IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 July 2022, accepted 23 August 2022, date of publication 5 September 2022, date of current version 14 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204115

RESEARCH ARTICLE

Efficient Region-Based Skyline Computation for a Group of Users

GHONCHEH BABANEJAD DEHAKI[1], HAMIDAH IBRAHIM[1], (Member, IEEE), ALI A. ALWAN[2], FATIMAH SIDI[1], (Member, IEEE), NUR IZURA UDZIR[1], AND MA'ARUF MOHAMMED LAWAL[3]

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Selangor 43400, Malaysia
[2]School of Theoretical and Applied Science, Ramapo College of New Jersey, Mahwah, NJ 07430, USA
[3]Department of Computer Science, Faculty of Physical Sciences, Ahmadu Bello University, Zaria Kaduna State 810107, Nigeria

Corresponding author: Hamidah Ibrahim (hamidah.ibrahim@upm.edu.my)

This work was supported in part by the Ministry of Higher Education Malaysia through the Fundamental Research Grant Scheme under Grant FRGS/1/2020/ICT03/UPM/01/1, and in part by the Universiti Putra Malaysia.

ABSTRACT Recently, with the advancement of technology, ad-hoc meetings or impromptu gatherings are becoming more and more common. The meetings/gatherings which involve at least two people will require a specific physical point location that is useful or interesting to them, called *point of interest (PoI)*. These people might be residing at different locations; each with their own preferences which most likely to be different. Undoubtedly, given *n* people in a group, there will be *n* users' preferences. Finding a suitable *PoI* that meets these *n* users' preferences is not a straightforward task. Existing solutions that utilise skyline processing in discovering the best, most preferred objects in satisfying the preferences of a group of users within a predetermined area have shown acceptable results. However, these solutions have to be executed repeatedly for each query of a group of users since they do not exploit the possibilities that an area that has been visited by a group of users might be the area of interest of another group of users in the future. Inherently, they require rescanning the objects and recomputing the skylines of a previously visited region which is undoubtedly unwise and costly. This paper proposes the *Region-based Skyline for a Group of Users* (*RSGU*) and *Extended Region-based Skyline for a Group of Users (ERSGU)* frameworks which attempt to resolve the limitations of existing solutions. In this work, skylines objects are *PoIs* that are recommended to a group of users that are derived by analysing both the locations of the users, i.e. spatial attributes, as well as the spatial and non-spatial attributes of objects that are within a predetermined region of the group of users. Here, each region is partitioned into smaller units called fragments in such a way that overlapping areas between the currently and previously visited regions can be easily determined; while the results of computing the skylines of each fragment, known as fragment skylines, are saved to be utilised by the subsequent requests. Meanwhile, *ERSGU* has an additional feature in which the skylines derived for a group of users are not only based on the evaluation of the spatial and non-spatial attributes of the objects, but also the closeness of the objects to the desirable facilities or other interesting objects in the region. Undeniably, a *PoI* that is nearby to other attractions is appealing and worth the journey. Several experiments have been conducted and the results show that our proposed frameworks outperform the previous work with respect to CPU time.

INDEX TERMS Multi-criteria decision making, skyline queries, group of users, spatial and non-spatial attributes.

I. INTRODUCTION

Query processing is defined as the process of answering a query (request) to a database or an information system, which usually involves the following three main activities: (i) analysing and interpreting the query, (ii) searching through the space of stored data, and (iii) retrieving the results satisfying the query. The traditional query processing operates either by retrieving objects[1] from a collection of objects

The associate editor coordinating the review of this manuscript and approving it for publication was Vivek Kumar Sehgal.

[1]Without loss of generality, the term object is used throughout this paper to be in line with other research works in similar area. The terms data, data item, record, and tuple can also be used in this context.

that strictly satisfy each condition specified in the query or returning an empty result if otherwise. The recent developments in query processing attempt to relax these stringent requirements, by retrieving the best, most preferred objects from the collection according to the conditions specified in the query, also known as *user-defined preferences*. These preference queries employ preference evaluation techniques, have achieved significant success, as they are widely used in applications related to multi-criteria decision support. During the past two decades, several preference evaluation techniques have been introduced, among them are: top-k [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], skyline [12], [13], [14], [15], [16], [1], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], k-dominance [36], [26], [37], [38], [39], [40], [41] top-k dominating [3], [42], [5], [43], [44], [45], [46], [47], [48], and k-frequency [49].

The skyline operator introduced by [12] which is used to filter a set of interesting objects from a potentially large multi-dimensional set of objects by keeping only those objects that are not worse than any other, has been greatly explored in several studies in an attempt to accurately and efficiently solve problems of real-world applications that are related to decision support and decision making. It attempts to derive the best, most preferred set of objects known as *skylines objects* (or *skylines* in short) according to a set of evaluation criteria.[2] The process of computing skylines becomes more challenging when conflicting criteria are involved while the number of criteria to be considered is huge. A classic example is selecting a hotel for a holiday whereby hotels that are close to the beach are known to be expensive. While other criteria like facilities, rating, and service, are equally important, distance and price are examples of conflicting criteria.

Since the introduction of the skyline operator by [12], an abundance of skyline algorithms have been proposed for data processing in order to retrieve useful insights. These variants of skyline algorithms are introduced to deal with different characteristics of data, such as uncertain data [58], incomplete data [50], [51], [52], [53], encrypted data [54], and streaming data [55]; while others are based on the platform being considered like distributed database [51], cloud computing [38], road networks [56], and others [57], [58], [59]. Nonetheless, these skyline algorithms focus mainly on the optimisation problem of skyline computation for a given single user query. Since the location of the user is insignificant in these studies, hence, it is sufficient to derive skyline objects by considering only the attributes of these objects, also known as *non-spatial* attributes, as the evaluation criteria.

However, in real world scenario, due to the advancement of technology, ad-hoc meetings or impromptu gatherings are becoming more and more common [60], [61], [62], [63], [64], [65], [66], [67]. Intuitively, the meetings/gatherings will involve a group of people (at least two people) and they will have to decide on a specific physical point location that is useful or interesting to them, called *point of interest*[3] (henceforth referred to as *PoI*). Some examples of *PoI* are restaurants, hotels, cafes, etc. These people might be residing at different locations; each with their own preferences which are most likely to be different. Undoubtedly, given n people in a group, there will be n users' queries.[4] The existing skyline algorithms are unsuited for such a scenario for two main reasons: (i) they cater a user's preferences at a time (single user query) and (ii) they deal only with the non-spatial attributes of the objects. While, deriving the skyline objects of a group of users that are located at different locations, both the spatial and non-spatial attributes of the objects need to be considered. Thus, identifying objects that best meet the preferences of a group of users is crucial and challenging.

The following scenarios typify the samples of situations considered in this paper.

Scenario I: Assuming several users whom are not close to each other would like to meet; hence a group is said to be formed and these users will have to decide on a specific *PoI* within a predetermined area[5] that is useful or interesting to them. There are many *PoIs* that they can choose. However, several criteria need to be considered before they decide on a specific *PoI* to visit. These include the location of the *PoI*, i.e. how far it is from the location of each user (spatial attribute), the opening hour, food, ticket price, rating, facilities provided, etc (non-spatial attributes). A *PoI* which is near to the users might not be the *PoI* that meets all the users' preferences. While a *PoI* which provides facilities that meet most of the users' preferences might be located far away from these users. Therefore, recommending a *PoI* to visit to these users is not a straightforward task as many criteria need to be considered. These include the location of each user (spatial attribute), the locations of the objects (spatial attribute), and the features of the objects (non-spatial attributes). It becomes more complicated when the number of users in the group is large, the objects to be analysed in the space (area) are high multi-dimensional while their number is huge. Meanwhile, an area that has been visited by a group of users might be the area of interest of another group of users in the future. Thus, it is essential to have a method that could find an object(s) that dominates other objects that best suits the preferences of a group of users with respect to both the spatial (location) and non-spatial attributes of the objects that are within a predetermined area. Also, it is hypothesised that utilising

[2] In this paper, the evaluation criteria used in determining the skyline objects are the attributes (the terms dimension and attribute are used interchangeably) of the objects.

[3] The term object of interest is also used that reflects the PoI that is saved as object in a data set.

[4] In this paper, the n users' queries are assumed to be distinct (reflecting different preferences) due to the fact that the skyline objects for a user might be different from another user as the area covered by each user's query might be different.

[5] Without loss of generality, the terms area, region, and space are used interchangeably throughout this paper.

the previous skyline computation results in identifying the skyline objects of the subsequent group of users can greatly reduce the skyline computation time.

Scenario II: Similar to the *Scenario* I described above, the group of users might be looking for a *PoI* to visit which is near to other useful or interesting facilities/objects (other types of *PoI*). Undeniably, a *PoI* that is nearby to other attractions is appealing and worth the journey. Therefore, besides considering the spatial attribute, i.e. the location of the objects, and the non-spatial attributes of the objects like opening hour, food, ticket price, rating, etc; another criterion needs to be established which is the distance between the *PoI* to other useful or interesting facilities/objects like mosque, cinema, hospital, etc. Thus, it is essential to have a method that could find an object(s) that dominates other objects that best suits the preferences of a group of users with respect to both the spatial (location) and non-spatial attributes of the objects as well as how close the object is to other useful or interesting facilities/objects that are within a predetermined area.

This paper takes the challenge to solve the problem associated to identifying skyline objects for a group of users whom intend to have a meeting/gathering. Two different solutions are proposed, each handling a different scenario as described in *Scenario* I and *Scenario* II. Generally, each solution will make use of the spatial attribute of the users, as well as the spatial and non-spatial attributes of the objects. In general, the main contributions of this work are briefly described as follows:

- We have formally introduced the problem of computing skylines of a group of users and justify the significance of addressing the problem.
- We have proposed an efficient solution, named *Region-based Skyline for a Group of Users* (*RSGU*) framework that is designed for processing the skyline queries of a given group of users by considering both the locations of the users in the group, as well as the spatial and non-spatial attributes of the objects that are within a predetermined region of the group of users; with two main aims that are (i) avoiding the process of rescanning the set of objects within a predetermined region that is known to have been previously visited by a group of users and (ii) avoiding the recomputation of skylines of a set of objects within a predetermined region that has been analysed in earlier computations of previously visited group of users.
- We have proposed the *Extended Region-based Skyline for a Group of Users (ERSGU)* framework, an enhancement of the *RSGU* framework, which has similar aims as *RSGU* with an additional feature in which the closeness of the objects to the other desirable facilities or interesting objects in the region are taken into consideration in the derivation of skyline objects for a group of users.
- We have conducted extensive experiments to prove *RSGU*'s and *ERSGU*'s capabilities in deriving the skyline objects for a group of users.

The rest of the paper is structured as follows. In Section II, the previous works that are related to computing skylines *for a single user* as well as for a *group of users* are presented. In Section III, the necessary definitions and notations, which are used throughout the paper, are set out. Section IV and Section V elaborate our proposed frameworks, *RSGU* and *ERSGU* respectively, that are purposedly designed for handling the computation of skyline objects for a group of users. A running database example is also given to clarify the phases of the proposed frameworks. The experimental results are demonstrated in Section VI. Conclusion and further research direction are depicted in the final section, Section VII.

II. RELATED WORKS

Since the introduction of the skyline operator by [12] many variants of skyline algorithms have been proposed. Although the ultimate goal of these algorithms is to derive the best, most preferred objects from a multi-dimensional set of objects, each of them tackled a slightly different issue. We categorised these skyline algorithms into two main categories, namely: skyline algorithms for a single user and skyline algorithms for a group of users.

Skyline algorithms for a single user – Generally, these skyline algorithms attempt to optimise the process of filtering the best, most preferred objects from a potentially large multi-dimensional set of objects. These algorithms aim at reducing the processing time by reducing the search space as small as possible. Thus, ensuring that only the set of objects that may potentially be the skylines is analysed. In this category, users' queries are assumed to have the same objective function; hence users are assumed to have the same preferences.

Among the earlier and most cited skyline algorithms in the literature are *Block Nested Loop* (*BNL*) [12], *Divide-and-Conquer* (*D&C*) [12], *Linear Elimination Sort for Skyline* (*LESS*) [68], *Branch and Bound Skyline* (*BBS*) [69], *SkyCube* [12], and *Sort and Limit Skyline algorithm* (*SaLSa*) [70]. Recently, several skyline algorithms have been proposed that attempt not only to resolve the optimisation problem but also issues related to the uncertainty of data; which is defined as the degree to which data are inaccurate, imprecise, untrusted, unknown or incomplete. These include among others *ISkyline* [22], *sorting-based bucket skyline* [71], *Incoskyline* [72], *Jincoskyline* [73], and *OIS* [74] that handle the issues of incompleteness of data. The incompleteness of data leads to the loss of *transitivity property* of skyline technique. It also leads to *cyclic dominance* between the objects as some objects are incomparable to each other that results in no object can be considered as skyline. Meanwhile, *probabilistic skyline model* [75], *τ-Skyline* [76], *SkyQUD* [77], [78], [79], [80] and *SQUiD* [71] focus on the challenges in computing skyline queries for uncertain database. Here, the exact values of the objects are not known at the point of processing. Consequently, one cannot derive the exact skyline but can only compute the probability of an object being a skyline member. On the other hand, the works by [81] and [82] attempt to solve the issues related to uncertain data in a data stream. Processing such data is challenging due to the objects in the stream arrive online and data streams are potentially unbounded in size. Besides, the work by [52] focuses on dynamic database. Nonetheless, these skyline algorithms are specifically designed to cater only a single user query, i.e. only a single user's preferences is considered in the skyline computation.

Skyline algorithms for a group of users – These skyline algorithms compute the skyline objects of a group of users from a potentially large multi-dimensional set of objects. As we assume that the objects are static, hence we further elaborate only those works that are similar to our intention. To the best of our knowledge the only works that contribute to skyline queries for a group of users are the works done by [60], [61], and [63]. In processing spatial skyline query for a group of users, two algorithms are proposed by [60], namely: B^2S^2 and VS^2. Both algorithms assumed that the user points are static. The B^2S^2 algorithm utilises the *R*-tree while the VS^2 algorithm utilises the Voronoi diagram. Then, [61] proposed the VCS^2 algorithm which enhanced the work by [60]. VCS^2 algorithm aims at processing skyline query by taking into consideration the movements of the users. However, VCS^2 only calculates the last location of the users and does not consider the changes of locations to prevent recalculation of the skylines. In [63], the authors proposed the *VR* algorithm, that combined two data structures as used in [61], *R*-tree and Voronoi, in order to find spatial skylines for a group of user points. In their work, both the user points and objects are considered static. While the spatial and non-spatial attributes of the objects are analysed to find the skylines. Meanwhile, our previous solution, *SGMU* [83], is designed with the main aim to continuously derive skylines for a group of mobile users.

Although [60], [61], [63], [83] considered the spatial attributes of the group of users in determining the skylines, but there is no attempt made to avoid rescanning of objects of previously visited regions and simultaneously avoid repeating the process of pairwise comparisons among the objects.

III. DEFINITIONS AND NOTATIONS

In this section, we present the necessary definitions and introduce the notations that are used throughout this paper. First, we give the definitions that are related to *RSGU*. This is then followed with definitions of *RSGU* that are modified/extended to suit with the *ERSGU*'s solution. Examples are provided where necessary to further clarify the definitions. A formal definition of the problem addressed by each solution is then put forward at the end of each section.

A. PRELIMINARIES OF RSGU

To clarify the concepts and steps proposed in this work, the following sample of data is used. Table 1(a) and Table 1(b) present the spatial attribute (*Location*) of the users of group a, G_a, and group b, G_b, respectively. Here, the request submitted by G_a is assumed at time t_a, while the request submitted by G_b is at time t_b where $t_a < t_b$. Table 2 presents the spatial (*Location*) and non-spatial (*Rate*, *Price*) attributes of the objects. For the non-spatial attributes, it is assumed that higher rate and lower price are preferable.

TABLE 1. The spatial attribute of the users.

ID	Location
u_1	(8, 8)
u_2	(14, 16)
u_3	(2, 5)

(a) Group a, G_a

ID	Location
u_1	(5, 8)
u_2	(10, 10)
u_3	(18, 10)

(b) Group b, G_b

TABLE 2. The spatial and non-spatial attributes of the objects.

Restaurant	Location	Rate	Price	Restaurant	Location	Rate	Price
o_1	(2, 3)	3	70	o_{24}	(4, 13.3)	3	75
o_2	(3, 4)	4	65	o_{25}	(7, 13)	1	90
o_3	(3, 1)	5	80	o_{26}	(16, 15)	2	86
o_4	(7, 1.7)	2	75	o_{27}	(20, 14)	5	80
o_5	(6, 5)	3	65	o_{28}	(23, 20)	3	60
o_6	(7, 7)	5	70	o_{29}	(21, 21)	5	62
o_7	(9, 8)	1	80	o_{30}	(17, 23)	4	95
o_8	(8, 9.7)	2	85	o_{31}	(14, 20)	2	65
o_9	(7, 11)	4	73	o_{32}	(13, 18)	2	55
o_{10}	(10, 5)	3	50	o_{33}	(10, 19)	3	70
o_{11}	(10.7, 6)	1	65	o_{34}	(1, 16)	4	62
o_{12}	(15, 2)	2	80	o_{35}	(3, 22)	4	81
o_{13}	(17, 1)	5	105	o_{36}	(7, 20)	3	90
o_{14}	(22, 4.7)	4	90	o_{37}	(24, 15)	2	66
o_{15}	(17, 5.7)	3	85	o_{38}	(-3, -1)	1	57
o_{16}	(20, 7)	4	90	o_{39}	(-1, 7)	1	61
o_{17}	(23, 9)	1	55	o_{40}	(10.3,13)	4	71
o_{18}	(16, 8)	2	54	o_{41}	(-4, 4)	3	98
o_{19}	(14, 10)	4	80	o_{42}	(8, -2)	2	58
o_{20}	(11, 9.7)	5	56	o_{43}	(8, 18)	2	85
o_{21}	(4, 10)	3	67	o_{44}	(-2, 10)	4	70
o_{22}	(2, 12)	5	100	o_{45}	(3, -1)	5	80
o_{23}	(3, 13)	4	74				

Given a data set $D = < R, U, O >$, where $U = \{u_1, u_2, \ldots, u_n\}$ is a list of n users, $O = \{o_1, o_2, \ldots, o_m\}$ is a list of m objects, and $R =< A_S, A_N >$ where A_S representing a spatial attribute while $A_N = \{d_1, d_2, \ldots, d_l\}$ is a set of non-spatial attributes. The following definitions defined the properties of a user and an object as used in this work.

Definition 1 (Property of a User): Each user, $u_i \in U$, is associated with a spatial attribute which represents the location of the user at a time, t. This is denoted as $u_i(x_i, y_i)$. For instance, $u_1(8, 8)$ of Table 1(a) denotes the location of user u_1 at time t_a.

Definition 2 (Properties of an Object): Each object $o_j \in O$ has two main elements denoted by $o_j = (s_j, ns_j)$ where s_j is the value of spatial attribute (location), A_S, and $ns_j = \{o_j.d_1, o_j.d_2, \ldots, o_j.d_l\}$ is a set of values of non-spatial attributes, A_N, associated to o_j. The location of an object $o_j \in O$ is denoted as $o_j(x_j, y_j)$. As each object $o_j \in O$ is assumed to be static, thus the location of the object is fixed regardless the changes in time. Hence, $o_j = (s_j, ns_j)$ can be written as $o_j = ((x_j, y_j), \{o_j.d_1, o_j.d_2, \ldots, o_j.d_l)\}$. For instance, the object o_1 of Table 2 can be written as $o_1 = ((2, 3), \{3, 70\})$.

The following definitions defined the notion of dominance in this work.

Definition 3 (Dominance[6]): Given two objects $o_i = (s_i, ns_i)$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to dominate o_j (denoted by $o_i \prec o_j$) if and only if both of the following conditions hold: (1) o_i non-spatially dominates o_j ($o_i \prec_{ns} o_j$) and (2) o_i spatially dominates o_j ($o_i \prec_s o_j$).

Definition 4 (Non-spatial Dominance): Given two objects $o_i = (s_i, ns_i)$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to non-spatially dominate o_j (denoted by $o_i \prec_{ns} o_j$) if and only if o_i is no worse than (in this definition, greater value is preferable) o_j in all the non-spatial attributes, A_N. This is formally written as follows: $o_i \prec_{ns} o_j$ if and only if $\forall d_k \in A_N, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in A_N, o_i.d_l > o_j.d_l$. For instance, given $o_6 = ((7, 7), \{5, 70\})$ and $o_{12} = ((15, 2), \{2, 80\})$, $o_6 \prec_{ns} o_{12}$ since o_6 is better than o_{12} in both the dimensions *Rate* and *Price*; with the assumption that higher rate and lower price are preferable.

Definition 5 (Spatial Dominance): Given two objects $o_i = (s_i, ns_i)$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to spatially dominate o_j (denoted by $o_i \prec_s o_j$) if and only if for every user $u_k \in U$, the distance between o_i and u_k, $dist(o_i, u_k)$, is no worse than the distance between o_j and u_k, $dist(o_j, u_k)$. This is formally written as follows: $o_i \prec_s o_j$ if and only if $\forall u_k \in U, dist(o_i, u_k) \leq dist(o_j, u_k) \wedge \exists u_l \in U, dist(o_i, u_l) < dist(o_j, u_l)$. For instance, the distances between $o_1 = ((2, 3), \{3, 70\})$ and u_1, u_2, and u_3 of group G_a are 7.81, 17.69, and 2, respectively; while the distances between $o_2 = ((3, 4), \{4, 65\})$ and u_1, u_2, and u_3 of group G_a are 6.4, 16.27, and 1.41, respectively. Thus, $o_2 \prec_s o_1$.

Definition 6 (Dominance in a Space): Given a bounded space, S (region, *MBR*, fragment, area, polygon, etc.), and two objects $o_i = (s_i, ns_i)$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$ in S, o_i is said to dominate o_j (denoted by $o_i \prec o_j$) in S if and only if (1) o_i non-spatially dominates o_j ($o_i \prec_{ns} o_j$) in S and (2) o_i spatially dominates o_j ($o_i \prec_s o_j$) in S.

Definition 7 (Skylines of a Space): An object $o_i \in O$ in a space S is a skyline of S if there are no other objects $o_j \in O$ where $i \neq j$ in the space S that dominates o_i. In this paper, Sky_{G_p} is used to denote the skyline set for the group G_p of a given space S.

Based on the above definitions, the problem that is tackled by *RSGU* is formulated as follows:

Given a group of users, $G_p = \{u_1, u_2, \ldots, u_p\}$, where $G_p \subset U$, and the candidate skylines of G_p in region R_p denoted as CS_{G_p}. Find the skylines of a group of users $G_q = \{u_1, u_2, \ldots, u_q\}$ in region R_q, i.e. CS_{G_q}, where $G_q \subset U$, $G_q \neq G_p$, and $R_q \cap R_p \neq \emptyset$ by utilising CS_{G_p} that has been derived for G_p. This is depicted in Fig. 1 where the area covered to compute the skylines for G_q that falls in the region S_{G_q} can be reduced to the area defined by $S_{G_q} - S_{G_p}$, while the results of skyline computation that have been performed earlier over the area $S_{G_p} \cap S_{G_q}$ for G_p can be avoided by simply utilising the obtained results derived for G_p, i.e. CS_{G_p}.

[6]Without loss of generality, the definition is applicable for a given bounded space, S, i.e. O is a set of objects in the space S. Similar note applies for Definition 4 and Definition 5.

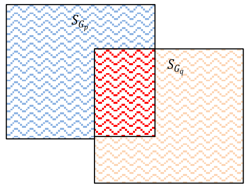


FIGURE 1. The reduction area in deriving skylines for a group of users.

TABLE 3. The spatial attribute of the interesting objects.

ID	Location
IN_{1-1}	(10, 7)
IN_{1-2}	(5, 12)
IN_{1-3}	(3, 15)
IN_{2-1}	(2.5, 9)
IN_{2-2}	(8, 4)
IN_{2-3}	(11.5, 3)
IN_{3-1}	(16, 10)
IN_{3-2}	(4, 11)
IN_{3-3}	(9, 6.5)

B. PRELIMINARIES OF ERSGU

Since *ERSGU* is an extended framework of *RSGU*, thus the concepts, terms, and notations introduced and clarified in Part *A* above are applied here. Also, the sample of data given in Table 1 and Table 2 will be referred to as example to clarify the steps of *ERSGU*. Therefore, the definitions of *property of a user*, *properties of an object*, *non-spatial dominance*, *spatial dominance*, *non-spatial dominance of the fragment* F_k, and *candidate skylines of the fragment* F_k are omitted here; readers can easily refer to their definitions in Part *A*. Nevertheless, three new definitions are included that are *Definition* 10 *Property of an Interesting Object*, *Definition* 11 *Closest Property of a t Type of Interesting Objects*, and *Definition* 13 *p-Closest Dominance*. These are further elaborated below.

Given a set of p distinct types of interesting objects, $IN = \{IN_1, IN_2, \ldots, IN_p\}$, where $IN_t = \{IN_{t-1}, IN_{t-2}, \ldots, IN_{t-n}\}$ is a list of n interesting objects of type t. The notation IN_{t-l} is used to denote the l-th interesting object of type t.

Definition 10 (Property of an Interesting Object): Each interesting object, $IN_{t-l} \in IN_t$, is associated with a spatial attribute which represents the location of the interesting object. This is denoted as $IN_{t-l}(x_l, y_l)$.

Table 3 presents samples of interesting objects that are used throughout this paper. There are three types of interesting objects with each type having three objects. The IN_{2-3} in Table 3 represents the third object of type 2 with location (11.5, 3).

Definition 11 (Closest Property of a t Type of Interesting Objects): Given a set of t type of interesting objects, $IN_t = \{IN_{t-1}, IN_{t-2}, \ldots, IN_{t-n}\}$, IN_{t-i} is said to be the closest interesting object to an object $o_j \in O$ if and only if $min(dist(o_j, IN_{t-1}), dist(o_j, IN_{t-2}), \ldots, dist(o_j, IN_{t-n})) = dist(o_j, IN_{t-i})$. Therefore, each object $o_j \in O$ is associated with p interesting objects that are closest to it, with their distances captured.

The *Definition* 3 *Dominance*, *Definition* 6 *Dominance in a Space*, and *Definition* 7 *Skylines of a Space* defined in Part *A* are extended to incorporate the additional condition on interesting objects. These are reflected in *Definition* 12 *Dominance*, *Definition* 14 *Dominance in a Space*, and *Definition* 15 *Skylines of a Space*, respectively.

Definition 12 (Dominance): Given two objects $o_i = (s_i, ns_i)$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to dominate o_j (denoted by $o_i \prec o_j$) if and only if the following conditions hold: (1) o_i non-spatially dominates o_j ($o_i \prec_{ns} o_j$), (2) o_i spatially dominates o_j ($o_i \prec_s o_j$), and (3) o_i dominates o_j with p-closest interesting objects ($o_i \prec_{s-IN} o_j$).

The definition of *non-spatially dominate* (condition (1)) is as given in *Definition* 5; while the definition of *spatially dominate* (condition (2)) is as given in *Definition* 6 of Part *A*. Meanwhile, *Definition* 13 *p-Closest Dominance* is introduced to cater the condition (3) defined in *Definition* 12 *Dominance*.

Definition 13 (p-Closest Dominance): Given two objects $o_i = (s_i, ns_i)$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to dominate o_j with p interesting objects (denoted by $o_i \prec_{s-IN} o_j$) if and only if o_i is no worse than o_j in all the p types of interesting objects. This is formally written as follows: $o_i \prec_{s-IN} o_j$ if and only if $\forall t \in p, o_i.t \leq o_j.t \wedge \exists s \in p, o_i.s < o_j.s$.

Definition 14 (Dominance in a Space): Given a bounded space, S (region, *MBR*, fragment, area, polygon, etc.), and two objects $o_i = (s_i, ns_i)$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$ in S, o_i is said to dominate o_j (denoted by $o_i \prec o_j$) in S if and only if (1) o_i non-spatially dominates o_j ($o_i \prec_{ns} o_j$) in S, (2) o_i spatially dominates o_j ($o_i \prec_s o_j$) in S, and o_i dominates o_j with p-closest interesting objects ($o_i \prec_{s-IN} o_j$) in S.

Definition 15 (Skylines of a Space): An object $o_i \in O$ in a space S is a skyline of S if there are no other objects $o_j \in O$ where $i \neq j$ in the space S that dominates o_i. In this paper, Sky_{G_p} is used to denote the skyline set for the group G_p of a given space S.

Based on the above definitions, the problem that is tackled by *ERSGU* is formulated as follows:

Given a group of users, $G_p = \{u_1, u_2, \ldots, u_p\}$, where $G_p \subset U$, and the candidate skylines of G_p in region R_p denoted as CS_{G_p}. Find the skylines of a group of users $G_q = \{u_1, u_2, \ldots, u_q\}$ in region R_q, i.e. CS_{G_q}, where $G_q \subset U$, $G_q \neq G_p$, and $R_q \cap R_p \neq \emptyset$ by utilising CS_{G_p} that has been derived for G_p. Note that the CS_{G_p} has been derived based on the objects' spatial and non-spatial attributes as well as their p-closest interesting objects. The area covered to compute the skylines for G_q that falls in the region S_{G_q} can be reduced to the area defined by $S_{G_q} - S_{G_p}$, while the results of skyline computation that have been performed earlier over the area $S_{G_p} \cap S_{G_q}$ for G_p can be avoided by simply utilising the obtained results derived for G_p, i.e. CS_{G_p}.

IV. THE RSGU FRAMEWORK

This section presents the *Region-based Skyline for a Group of Users* (*RSGU*) framework that is mainly proposed for processing the skyline queries of a given group of users; with two main aims that are (i) avoiding the process of rescanning the set of objects within a predetermined region that is known to have been previously visited by a group of users and (ii) avoiding the recomputation of skylines of a set of objects within a predetermined region that has been analysed in earlier computations of previously visited group of users. Consequently, the number of pairwise comparisons and skyline computation time can be greatly reduced. To achieve these aims, each region is partitioned into smaller units called fragments in such a way that overlapping areas between the currently and previously visited regions can be easily determined; hence rescanning the set of objects within these overlapping areas can be avoided. The results of computing the skylines of each fragment, known as fragment skylines, are saved to be utilised by the subsequent requests. This avoids the need to recompute the skylines of subsequent requests that fall within the same region. In this work, skylines objects are point of interests (*PoIs*) that are recommended to a group of users that are derived by analysing both the locations of the users, i.e. spatial attributes, as well as the spatial and non-spatial attributes of the set of objects that is within a predetermined region of the group of users.

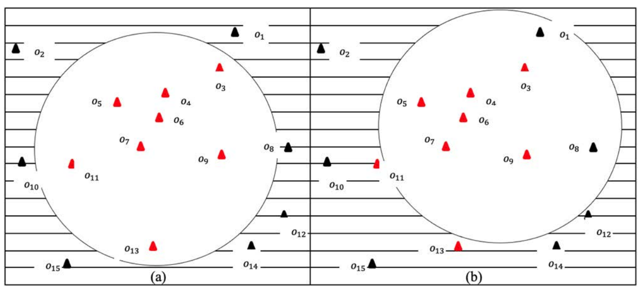


FIGURE 2. Example of (a) previously analysed region (b) current region.

Fig. 2 simulates a sample of situation considered in this paper. There are 15 distinct objects representing point of interests (*PoIs*). Fig. 2(a) presents the region that is derived based on the locations of a group of users while Fig. 2(b) presents the region that is derived based on the locations of a different group of users. For simplicity, these users are not depicted in the figure. The derivation of these regions is explained in the following sections. Here, the region presented in Fig. 2(b) is considered as the current visited region while Fig. 2(a) represents the previously visited region. As shown in Fig. 2(a) the set of objects that falls within the derived region is $\{o_3, o_4, o_5, o_6, o_7, o_9, o_{11}, o_{13}\}$. These objects are then compared to recognise the final skylines to be recommended to the group of users. Assume that the object o_7 is the skyline object. Meanwhile, based on the region depicted in Fig. 2(b), the set of objects that needs to be analysed in order to derive the skyline objects is $\{o_1, o_3, o_4, o_5, o_6, o_7, o_8, o_9\}$. From these figures, it is obvious that both regions cover similar area and contain several common objects, namely: $\{o_3, o_4, o_5, o_6, o_7$, and $o_9\}$. Attempting to rescan the objects of previously visited region and recompute the skylines of the regions (i.e. repeating the process of pairwise comparisons among objects) are undoubtedly unwise and costly. In this example, to derive the skylines of $\{o_1, o_3, o_4, o_5, o_6, o_7, o_8, o_9\}$ (Fig. 2(b)) would require pairwise comparisons to be performed between these objects; while the objects o_3, o_4, o_5, o_6, o_7, and o_9 have been compared while they are being analysed in identifying the skyline objects of the region presented in Fig. 2(a).

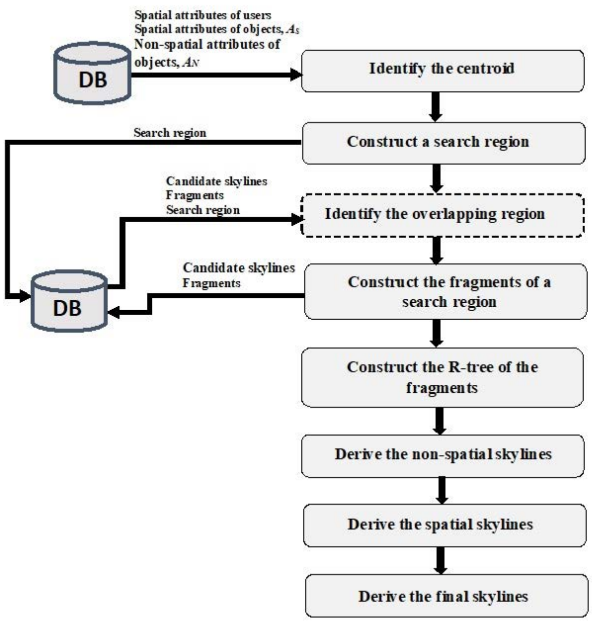


FIGURE 3. The Region-based Skyline for a Group of Users (RSGU) framework.

The *RSGU* framework is presented in Fig. 3. It consists of eight main steps that are: (1) Identify the centroid, (2) Construct a search region, (3) Identify the overlapping region, (4) Construct the fragments of a search region, (5) Construct the *R*-tree of the fragments, (6) Derive the non-spatial skylines, (7) Derive the spatial skylines, and (8) Derive the final skylines. Step (3) is conducted only when past computed skyline results of the fragments are available. Each of these steps is elaborated in the following sections.

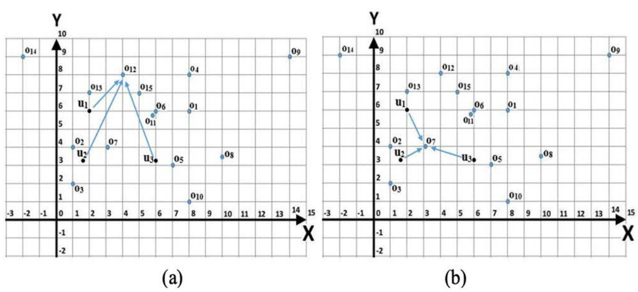


FIGURE 4. (a) The direction of movements of a group of users towards a point which is not the centre point (b) The direction of movements of a group of users towards a point which has the tendency to be the centre point.

A. IDENTIFY THE CENTROID

When a group of users, $G_p = \{u_1, u_2, \ldots, u_p\}$, decided to meet, there must be a point to guide the direction of their movements. Fig. 4 shows some examples of direction of movements of a group of users towards a targeted point. In Fig. 4(a), the users u_1, u_2, and u_3 are moving towards the object o_{12} while in Fig. 4(b), these users are moving towards the object o_7.

In this work, it is assumed that the group of users will move towards a point that has the tendency to be a center based on the users' locations. This point is called centroid and is denoted by $C_{G_p}(x_{G_p}, y_{G_p})$. The centroid of a given group of users, C_{G_p}, is determined using the following formula [84], [85]:

$$C_{G_p}(x_{G_p} = \frac{\sum_{i=1}^{n} xi}{n}, y_{G_p} = \frac{\sum_{i=1}^{n} yi}{n}) \quad (1)$$

where x_i is the x coordinate of user u_i location, y_i is the y coordinate of user u_i location, x_{G_p} is the average of the x coordinates of all users in the group G_p, and y_{G_p} is the average of the y coordinates of all users in the group G_p. Based on the example given in Table 2, the centroid of G_a is $C_{G_a}(8, 9.6)$.

B. CONSTRUCT A SEARCH REGION

The aim of constructing a search region is to limit the searching space to those spaces in which potential candidate skyline objects are derived. Hence, given a group of users, the searching space should include the regions of interest of all users in the group. This is achieved by: (1) identifying the search region for each user, S_{u_i} and (ii) identifying the search region given a group of users, S_{G_p}.

Identify the search region for each user, S_{u_i} – Since the centroid of a given group of users, say C_{G_p}, which is identified in the previous step does not necessarily contain an object, therefore the nearest object, o_n, to the centroid C_{G_p} will have to be determined. The nearest object is an object with the shortest Euclidean distance from the centroid, i.e. $\{o_n | o_n \in O \wedge \forall o_i \in O - \{o_n\}: Ed(C_{G_p}, o_n) < Ed(C_{G_p}, o_i)\}$ where Ed is the Euclidean distance function. Based on the example given in Table 1(a), the nearest object to the centroid of G_a, i.e. $C_{G_a}(8, 9.6)$, is $o_8(8, 9.7)$. The search region for a user, u_i, denoted as S_{u_i}, is the area bounded by a rectangle also known as the minimum bounding rectangle, MBR_{u_i}. The notation S_{u_i} is used to denote the search region of u_i while MBR_{u_i} is used in forming the S_{u_i}. The distance between a user, u_i, and the nearest object, o_n, denoted by $R_{u_i o_n}$, is calculated by the following equation:

$$R_{u_i o_n} = \sqrt{(x_{o_n} - x_i)^2 + (y_{o_n} - y_i)^2} \quad (2)$$

where x_i is the x coordinate of user u_i location, y_i is the y coordinate of user u_i location, x_{o_n} is the x coordinate of object o_n location, and y_{o_n} is the y coordinate of object o_n location. A *MBR* is formed based on four vertices as explained in the following: the vertex at the bottom left of the *MBR* is denoted by $bl = (x_{bl}, y_{bl})$; the vertex at the bottom right of the *MBR* is denoted by $br = (x_{br}, y_{br})$; the vertex at the top left of the

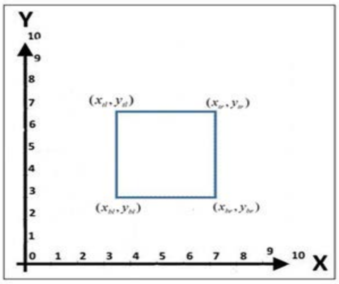


FIGURE 5. **Minimum bounding rectangle (MBR).**

MBR is denoted by $tl = (x_{tl}, y_{tl})$; and the vertex at the top right of the *MBR* is denoted by $tr = (x_{tr}, y_{tr})$. Fig. 5 depicts these notations. These vertices are calculated as follows:

$$bl = (x_i - R_{u_i o_n}, y_i - R_{u_i o_n})$$
$$br = (x_i + R_{u_i o_n}, y_i - R_{u_i o_n})$$
$$tl = (x_i - R_{u_i o_n}, y_i + R_{u_i o_n})$$
$$tr = \left(x_i + R_{u_i o_n}, y_i + R_{u_i o_n}\right)$$

Identify the search region given a group of users, S_{G_p} – This step is simply achieved by performing union on the search region of each user in the group, i.e. $S_{G_p} = \bigcup_{i=1}^{p} S_{u_i}$. An example of a search region $S_{G_a} = \bigcup_{i=1}^{3} S_{u_i}$ can be seen in Fig. 6. The search region constructed in this step is saved to be utilised later in identifying the overlapping region of subsequent requests.

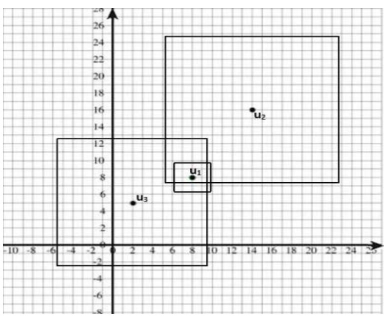


FIGURE 6. **The search region for a group of users.**

C. CONSTRUCT THE FRAGMENTS OF A SEARCH REGION

This step partitions the search region of a group of users, S_{G_p}, into *m* fragments (subspaces). Here, the vertices of the *MBR* associated to each S_{u_i} are analysed and sorted according to the *x*- and *y*-axes. The search region (space) is vertically fragmented based on the *x*-coordinates, while it is horizontally fragmented based on the *y*-coordinates. The *MRBs* formed within the S_{G_p} are the fragments of the region.

Objects that fall within each fragment are then identified. Given an object, $o_j(x_j, y_j)$, and a fragment, F_k, with $bl(x_l, y_b)$, $br(x_r, y_b)$, $tl(x_l, y_t)$, and $tr(x_r, y_t)$, the following cases are identified:

(a) If $x_l < x_j < x_r$ and $y_b < y_j < y_t$, then the object $o_j(x_j, y_j)$ is said to fall within the boundary of fragment, F_k.
(b) If $x_j = x_l$ or $x_j = x_r$ or $y_j = y_b$ or $y_j = y_t$, then the object $o_j(x_j, y_j)$ is said to intersect with the boundary of fragment, F_k.
(c) Objects that do not meet the above two cases are objects that are outside the boundary of fragment, F_k.

Further, utilising the non-spatial dominance testing given in *Definition* 8, an extension to the *Definition* 4, over the set of objects that satisfies the cases (a) or (b) above, denoted by O_{F_k}, the non-spatial candidate skylines of a fragment are determined, CS_{nsF_k}, as defined by *Definition* 9.

Definition 8 (Non-Spatial Dominance of the Fragment F_k): Given two objects $o_i = (s_i, ns_i) \in O_{F_k}$ and $o_j = (s_j, ns_j) \in O_{F_k}$ where $i \neq j$, o_i is said to non-spatially dominate o_j (denoted by $o_i \prec_{ns} o_j$) if and only if o_i is no worse than (in this definition, greater value is preferable) o_j in all the non-spatial attributes, A_N. This is formally written as follows: $o_i \prec_{ns} o_j$ if and only if $\forall d_k \in A_N, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in A_N$, $o_i.d_l > o_j.d_l$.

Definition 9 (Candidate Skylines of the Fragment F_k): An object $o_i \in O_{F_k}$ in a space F_k is a non-spatial candidate skyline of F_k if there are no other objects $o_j \in O_{F_k}$ where $i \neq j$ in the space F_k that non-spatially dominates o_i.

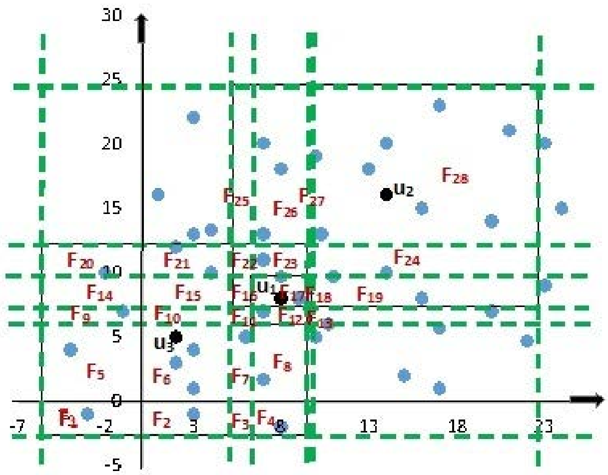


FIGURE 7. **The fragments derived based on the S_{G_a} given in Fig. 6.**

This will avoid rescanning the objects of the region and repeating the process of pairwise comparisons among the objects during the skyline computation of subsequent skyline queries. Fig. 7 presents the fragments constructed based on the S_{G_a} given in Fig. 6. The *x*-coordinates = {-5.6, 0, 5.3, 6.3, 9.6, 9.7, 22.7} and *y*-coordinates = {-2.6, 0, 6.3, 7.3, 9.7, 12.6, 24.7}. Altogether there are 28 fragments; some samples are given in Table 4.

D. CONSTRUCT THE R-TREE OF THE FRAGMENTS

The aim of constructing the *R*-tree of the fragments is to reduce the searching process time. An *R*-tree is a classified data structure and it is used for dynamic classification of a set

TABLE 4. Sample of fragments and their associated candidate skylines.

x Coordinate (x_l, x_r)	y Coordinate (y_b, y_t)	$bl(x_l, y_b)$	$br(x_r, y_b)$	$tl(x_l, y_t)$	$tr(x_r, y_t)$	Fragment, F_k	Objects	Candidate skylines, CS_{nsF_k}
-5.6, 0	-2.6, 0	-5.6, -2.6	0, -2.6	-5.6, 0	0, 0	F_1	o_{38}	o_{38}
0, 5.3	-2.6, 0	0, -2.6	5.3, -2.6	0, 0	5.3, 0	F_2	o_{45}	o_{45}
5.3, 6.3	-2.6, 0	5.3, -2.6	6.3, -2.6	5.3, 0	6.3, 0	F_3	-	-
6.3, 9.6	-2.6, 0	6.3, -2.6	9.6, -2.6	6.3, 0	9.6, 0	F_4	o_{42}	o_{42}
…	…	…	…	…	…	…	…	…
0, 5.3	0, 6.3	0, 0	5.3, 0	0, 6.3	5.3, 6.3	F_6	o_1, o_2, o_3	o_2, o_3
…	…	…	…	…	…	…	…	…
9.7, 22.7	7.3, 24.7	9.7, 7.3	22.7, 7.3	9.7, 24.7	22.7, 24.7	F_{28}	$o_{26}, o_{27}, o_{29}, o_{30}, o_{31}, o_{32}, o_{33}, o_{40}$	o_{29}, o_{32}

of d-dimensional coordination, rectangles or objects demonstrating them by the minimum bounding d-dimensional rectangles. Each node of the R-tree relates to the *MBR* that confines its children. An R-tree of order (m, M) considering each leaf node can have up to M entries, while the minimum permitted number of entries is $m <= M/2$. All leaf nodes of the R-tree are at the same level. The R-tree is constructed based on the algorithms proposed by [84]. In this work, the following algorithms are utilised during the construction of the R-tree: *Search*, *Insert*, *ChooseLeaf*, *SplitNode*, and *AdjustTree*.

Based on Fig. 7 the *MBRs* constructed are as shown in Fig. 8 while Fig. 9 shows the R-tree derived based on the *MBRs* of the fragments.

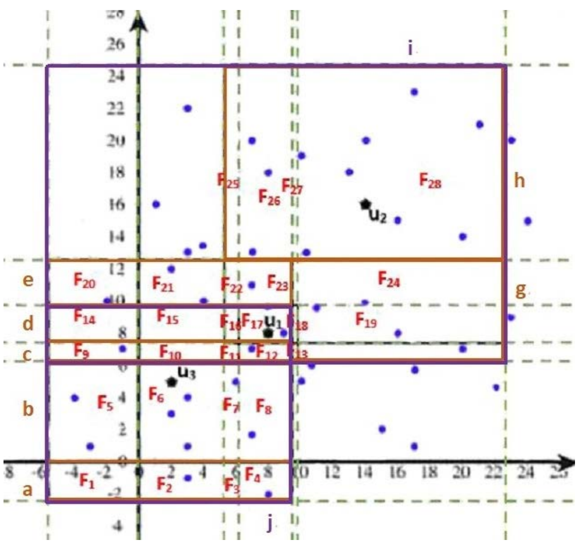


FIGURE 8. The MBRs derived based on the fragments of the S_{G_a} given in Fig. 7.

E. DERIVE THE NON-SPATIAL SKYLINES

This step performs the non-spatial dominance testing given in *Definition* 8 *Non-spatial Dominance of the Fragment* F_k towards the CS_{nsF_i} lists derived in the previous step, *Construct the Fragments of a Search Region*, presented in Part *C*

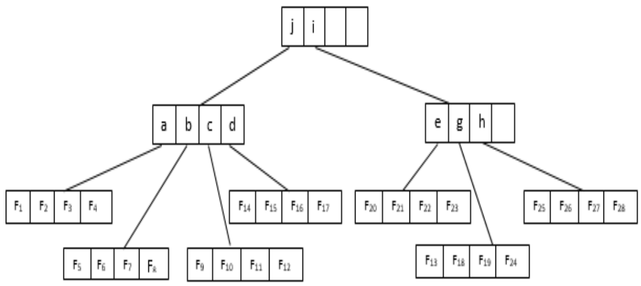


FIGURE 9. The R-tree derived based on the MBRs of the fragments given in Fig. 8.

to generate the non-spatial skylines of a given group of users. In other words, the pairwise comparisons are only performed between objects that are the candidate skylines of a fragment. The objects that non-spatially dominate the other objects, given the CS_{nsF_i} lists where $i = \{1, 2, \ldots, 28\}$ in Table 4 are o_{18} and o_{20}, thus $Sky_{nsG_a} = \{o_{18}, o_{20}\}$.

F. DERIVE THE SPATIAL SKYLINES

This step applies the spatial dominance testing given in *Definition* 5 *Spatial Dominance* towards the CS_{nsF_i} lists. First, the distance between each object, o_i, and each user, u_j, is determined, denoted as $o_i - u_j$. Given a group of l users, there will be l values of distances with regard to the object o_i, i.e. $o_i - u_1, o_i - u_2, \ldots, o_i - u_l$. These values are treated as the values of dimensions to be used in the spatial skyline computation. Then, the total distance between an object, o_i, to each user, u_j, in the group of users is calculated and saved into a parameter named *Sum Distance*$-o_i$, i.e. *Sum Distance*$-o_i = \sum_{p=1}^{l} o_i - u_p$. The value of *Sum Distance* $- o_i$ is used as a selection criterion in determining the object that should be considered in each iteration of the spatial skyline computation. The smallest value of *Sum Distance* $- o_i$ indirectly indicates that most users in the group are close to the object o_i and has more chances to dominate the other objects.

An example is shown in Table 5 where the attributes $o_i - u_1$, $o_i - u_2$, and $o_i - u_3$ are the distances of each object,

TABLE 5. The distance and sum distance of each object in $CS_{ns_{F_t}}$.

Restaurant	$o_i - u_1$	$o_i - u_2$	$o_i - u_3$	$Sum\ Distance - o_i$
o_1	7.81	17.69	2	27.5
o_2	6.4	16.27	1.41	24.08
o_3	8.6	18.6	4.12	31.32
o_4	6.37	15.92	5.99	28.28
o_5	3.6	13.6	4	21.2
o_6	1.41	11.4	5.38	18.19
o_7	1	9.43	7.61	18.04
o_8	1.7	8.7	7.62	18.02
o_9	3.16	8.6	7.81	19.57
o_{18}	8	8.24	14.31	30.55
o_{19}	6.32	6	13	25.32
o_{20}	3.44	6.97	10.15	20.56
o_{21}	4.47	11.66	5.38	21.51
o_{22}	7.21	12.64	7	26.85
o_{25}	5.09	7.61	9.43	22.13
o_{26}	10.63	2.23	17.2	30.06
o_{27}	13.41	6.32	20.12	39.85
o_{29}	18.38	8.6	24.83	51.81
o_{30}	18.6	7.61	23.43	49.64
o_{31}	13.41	4	19.2	36.61
o_{32}	11.18	2.23	17.02	30.43
o_{33}	11.18	5	16.12	32.3
o_{36}	12.04	8.06	15.81	35.91
o_{38}	14.21	24.04	7.81	46.06
o_{39}	9.05	17.49	3.6	30.14
o_{40}	5.5	4.76	11.52	21.78
o_{41}	12.64	21.63	6.08	40.35
o_{42}	10	18.97	9.21	38.18
o_{43}	10	6.32	14.31	30.63
o_{44}	10.19	17.08	6.4	33.67
o_{45}	8.6	20.24	6.08	34.92

o_i, and each user u_1, u_2, and u_3, respectively; meanwhile the *Sum Distance* $- o_i$ attribute presents the total distance of an object to each user in the group of users. Here, o_8 will be the first object selected which is then followed by o_7. The objects that spatially dominate the other objects, given the $CS_{ns_{F_i}}$ lists in Table 4 are as listed in $Sky_{s_{G_a}} = \{o_2, o_5, o_6, o_7, o_8, o_9, o_{20}, o_{25}, o_{26}, o_{32}, o_{40}\}$.

G. DERIVE THE FINAL SKYLINES

This is the final step that combines the results produced in the steps presented in Parts *E* and *F* above. Based on *Definition* 7 *Skylines of a Space*, the final skylines for a given group G_i is given by, $Sky_{G_i} = Sky_{ns_{G_i}} \cup Sky_{s_{G_i}}$. Thus, the final skylines for the group G_a, $Sky_{G_a} = \{o_2, o_5, o_6, o_7, o_8, o_9, o_{18}, o_{20}, o_{25}, o_{26}, o_{32}, o_{40}\}$.

H. IDENTIFY THE OVERLAPPING REGION

This step constructs the overlapping region, O_R, between the search regions of two groups of users, say S_{G_i} and S_{G_j}. Here, it is assumed that the results of the skyline queries of a group of users, say G_i, have been derived. Thus, the overlapping region indicates that the region has been scanned and it is unwise to scan it again. Fig. 10 shows two search regions, S_{G_a}, the polygon with black border line and S_{G_b}, the polygon with red border line which represent the search region of group G_a and group G_b, respectively. If there are more than one search region that are available, $\{S_{G_i}, S_{G_k}, \ldots, S_{G_m}\}$, then the overlapping area between S_{G_j} and each of the available search region is analysed and the region with the highest percentage of overlapping area is selected in this step.

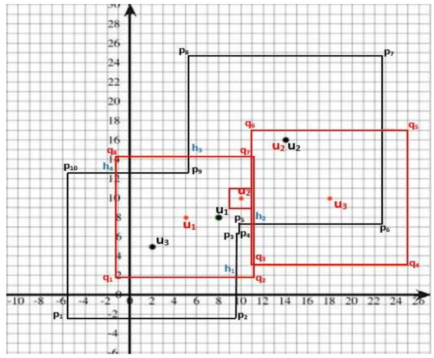


FIGURE 10. The overlapping region between S_{G_a} and S_{G_b}.

To identify the overlapping region, the following steps are performed:

(1) Get the polygon's vertices of S_{G_i}. Based on the example, $S_{G_a} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}\}$. Note that for simplicity, the coordinates of the vertices are omitted.

(2) Get the polygon's vertices of S_{G_j}. Based on the example, $S_{G_b} = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$.

(3) Get the vertices of S_{G_i} that are also in S_{G_j}. Based on the example, $I_{G_a-G_b} = \{p_3, p_4, p_5, p_6, p_9\}$.

(4) Get the vertices of S_{G_j} that are also in S_{G_i}. Based on the example, $I_{G_b-G_a} = \{q_1, q_6, q_7\}$.

(5) Get the coordinates where the edges of S_{G_i} and S_{G_j} meet. Based on the example, $H = \{h_1(9.6, 1.8), h_2(11.2, 7.3), h_3(22.7, 17), h_4(5.3, 14.2), h_5(-1.2.12.6)\}$.

(6) The overlapping region, O_R, is defined as a polygon derived based on the following vertices: $I_{G_a-G_b} \cup I_{G_b-G_a} \cup H$. Based on the example, $O_R = \{h_1, p_3, p_4, p_5, h_2, p_6, h_3, q_6, q_7, h_4, p_9, h_5, q_1\}$.

Once the O_R has been defined, the fragments derived in the earlier step are analysed. Those fragments that fall within the O_R; are retrieved together with their candidate skylines, $CS_{ns_{F_k}}$. To search for the fragments that are within the O_R, the *R*-tree constructed in Part *D* is traversed starting from the root node. The search is based on the depth-first search tree traversal algorithm. The following rules are applied: (i) If the visited node, v_n, is an internal node (example, the node with the a, b, c, d entries of Fig. 9), then each entry of the internal node, $e_i.v_n$, is examined (example a). If the entry (example a) overlaps with O_R, then rules (i) and (ii) are applied accordingly over the entry, $e_i.v_n$; (ii) If the visited node, v_n, is a leaf node (example, the node with the F_1, F_2, F_3, F_4 entries of Fig. 9), then each entry of the leaf node, $e_i.v_n$, is examined. If the entry overlaps with O_R, then it is one of the fragments that will be retrieved for further analysis. Hence, scanning this area is no longer necessary. While for the non-overlapping area, denoted as $\neg O_R$, the following steps as discussed above will be conducted: (4) *Construct the fragments of the non-overlapping region*, i.e. $\neg O_R = S_{G_j} - O_R$ (5 and 6) *Derive non-spatial skylines* and *spatial skylines*, respectively by considering both the lists CS_{O_R} and $CS_{\neg O_R}$, and (7) *Derive the final skylines*.

V. THE ERSGU FRAMEWORK

This section presents the *Extended Region-based Skyline for a Group of Users (ERSGU)* framework, an enhancement of the previous framework, *RSGU*, presented in Section IV. *RSGU* is designed for processing the skyline queries of a given group of users. *RSGU* attempts to avoid the process of rescanning the set of objects and the recomputation of skylines of a set of objects that is within a predetermined region that is known to have been previously visited by a group of users. As a result, the number of pairwise comparisons and skyline computation time can be greatly reduced. *ERSGU* has similar aims as *RSGU* with an additional feature. The skylines derived for a group of users by *ERSGU* are not only based on the evaluation of the spatial and non-spatial attributes of the objects that are within the predetermined region, but also the closeness of the objects to the desirable facilities or other interesting objects in the region. An example of object of interest is hotel while other desirable facilities/interesting objects are clinic, bus station, airport, etc. This would benefit the users, since they might want to visit a place where there are several other useful facilities or interesting objects (for simplicity, henceforth referred to as *interesting objects*) nearby. Hence, the skylines, which are the objects recommended to be visited by the group of users, are derived by analysing both the locations of the users, i.e. spatial attributes, as well as the spatial and non-spatial attributes of the objects along with the closeness of the objects to other interesting objects.

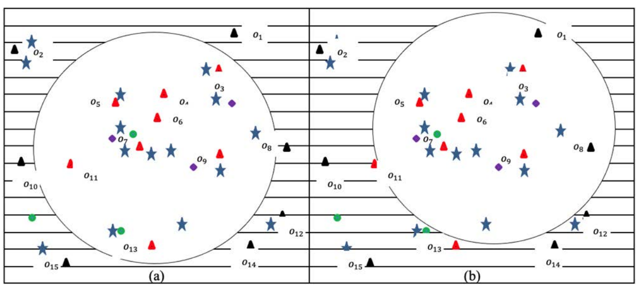


FIGURE 11. Example of (a) previously analysed region (b) current region with interesting objects/facilities.

Fig. 11 simulates a sample of situation considered in this paper which is similar to the sample of situation described in Fig. 2. Besides, the 15 distinct objects representing objects of interest (▲), several other interesting objects are also presented. The symbols (★), (●), and (◆) in the figure represent the distinct types of interesting objects. Fig. 11(a) presents the previously visited region that is derived based on the locations of a group of users in which the skylines of the region have been derived; while Fig. 11(b) presents the current visited region that is derived based on the locations of a different group of users in which the skylines are to be identified. As shown in Fig. 11(a) and Fig. 11(b) the sets of objects that fall within the derived regions are {o_3, o_4, o_5, o_6, o_7, o_9, o_{11}, o_{13}} and {o_1, o_3, o_4, o_5, o_6, o_7, o_8, o_9}, respectively. Each set of objects is then compared not only based on the objects' spatial and non-spatial attributes but also how close are they to the other interesting objects before the final skylines are recommended to the group of users. The object o_7 for instance is near to several interesting objects compared to the objects o_4 and o_6. From these figures, it is obvious that both regions cover similar area and contain several common objects, namely: o_3, o_4, o_5, o_6, o_7, and o_9. Hence, attempting to rescan the objects of previously visited region and recompute the skylines of the region (i.e. repeating the process of pairwise comparisons among objects) are undoubtedly unwise and costly. This is because not only the objects' spatial and non-spatial attributes will be analysed again but also all the other interesting objects that are close to the objects need to be reexplored.

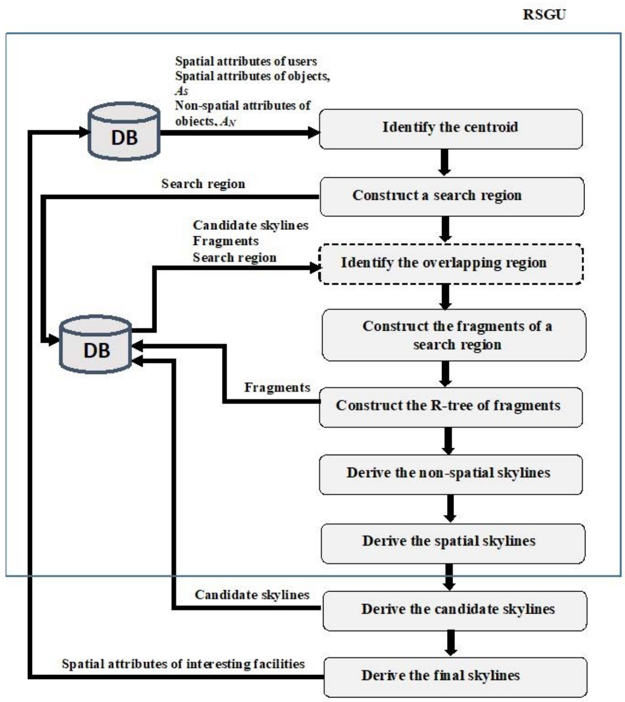


FIGURE 12. The extended region-based skyline for a Group of users (ERSGU) framework.

The *ERSGU* framework is presented in Fig. 12. It consists of nine main steps that are: (1) Identify the centroid, (2) Construct a search region, (3) Identify the overlapping region, (4) Construct the fragments of a search region, (5) Construct the *R*-tree of the fragments, (6) Derive the non-spatial skylines, (7) Derive the spatial skylines, (8) Derive the candidate skylines, and (9) Derive the final skylines. Steps (1) till (7) are the same steps as *RSGU*, while Step (8) *Derive the final skylines* of *RSGU* is renamed as (8) *Derive the candidate skylines* without any changes with regard to the step. Step (9) *Derive the final skylines* is the new step incorporated into *ERSGU*. Hence, only Step (8) and Step (9) of *ERSGU* will be further discussed while the earlier seven steps are as explained in Section IV.

A. DERIVE THE CANDIDATE SKYLINES

This step combines the results produced in the steps presented in Section IV Parts E and F. Based on *Definition* 7, the candidate skylines for a given group G_i is given by, $C - Sky_{G_i} = Sky_{ns_{G_i}} \cup Sky_{s_{G_i}}$. Thus, referring to the same example given in Section IV, the candidate skylines for the group G_a, $C - Sky_{G_a} = \{o_2, o_5, o_6, o_7, o_8, o_9, o_{18}, o_{20}, o_{25}, o_{26}, o_{32}, o_{40}\}$.

B. DERIVE THE FINAL SKYLINES

The final step of *ERSGU* attempts to derive the final skylines based on the list of candidate skylines produced in Step (8). The candidate skylines are those objects that are not dominated by any other objects based on the two conditions defined in *Definition* 12 *Dominance*. These conditions are (1) o_i non-spatially dominates o_j ($o_i \prec_{ns} o_j$) and (2) o_i spatially dominates o_j ($o_i \prec_s o_j$). The final skylines are derived based on the candidate skylines obtained above in which condition (3) of *Definition* 12 *Dominance* is satisfied, i.e. o_i dominates o_j with p-closest interesting objects ($o_i \prec_{s-IN} o_j$).

To clarify this step, consider the group of users, G_a, and its search region, S_{G_a}, as given in Table 1 and Fig. 6, respectively. Assume that the interesting objects that fall within the region of S_{G_a} are as presented in Fig. 13 with their detail locations presented in Table 3. In this example, the following are assumed:

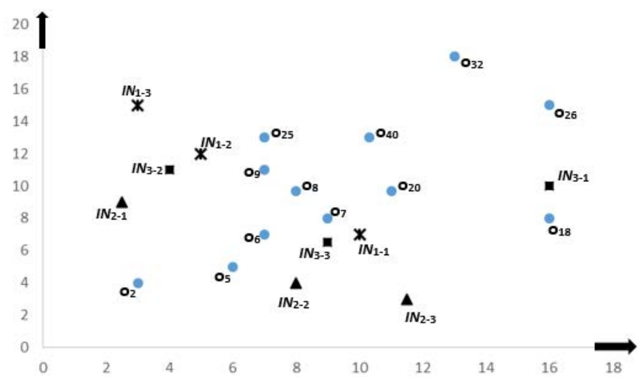


FIGURE 13. Locations of the interesting objects in the region S_{G_a}.

1) Only candidate skylines produced in Step (8) are presented in the figure, i.e. $C - Sky_{G_a} = \{o_2, o_5, o_6, o_7, o_8, o_9, o_{18}, o_{20}, o_{25}, o_{26}, o_{32}, o_{40}\}$.
These objects are denoted as circles.
2) There are three types of interesting objects symbolised by star (type 1), triangle (type 2), and rectangular (type 3).

Each type of interesting object has three objects. The notation IN_{t-l} is used to denote the l-th interesting object of type t. Hence, $IN_1 = \{IN_{1-1}, IN_{1-2}, IN_{1-3}\}$, $IN_2 = \{IN_{2-1}, IN_{2-2}, IN_{2-3}\}$, and $IN_3 = \{IN_{3-1}, IN_{3-2}, IN_{3-3}\}$ represent the set of interesting objects of type 1, 2, and 3, respectively. The locations of each of these objects are as given in Table 3.

TABLE 6. The distances between each $C - Sky_{Ga}$ and IN.

ID	IN_{1-1}	IN_{1-2}	IN_{1-3}	IN_{2-1}	IN_{2-2}	IN_{2-3}	IN_{3-1}	IN_{3-2}	IN_{3-3}
o_2	7.61	8.24	11	5.02	5	8.55	14.31	7.07	6.5
o_5	4.47	7.07	10.44	5.31	2.23	5.85	11.18	6.32	3.35
o_6	3	5.38	8.94	4.92	3.16	6.02	9.48	5	2.06
o_7	1.41	5.65	9.21	6.57	4.12	5.59	7.28	5.83	1.5
o_8	3.36	3.78	7.28	5.54	5.7	7.55	8	4.2	3.35
o_9	5	2.23	5.65	4.92	7.07	9.17	9.05	3	4.92
o_{18}	6.08	11.70	14.76	13.53	8.94	6.72	2	12.36	7.15
o_{20}	2.87	6.42	9.59	8.52	6.44	6.71	5	7.11	3.77
o_{25}	6.70	2.23	4.47	6.02	9.05	10.96	9.48	3.60	6.80
o_{26}	10	11.40	13	14.77	13.60	12.81	5	12.64	11.01
o_{32}	11.40	10	10.44	13.82	14.86	15.07	8.54	11.40	12.17
o_{40}	6	5.39	7.56	8.76	9.28	10.07	6.44	6.60	6.62

Given a set of candidate skylines for a group of users, $C - Sky_{G_i} = \{o_1, o_2, \ldots, o_y\}$ and a set of p distinct type of interesting objects, $IN = \{IN_1, IN_2, \ldots, IN_p\}$, the following steps are performed:

(a) For each $o_j \in C - Sky_{G_i}$ and for each set of t type of interesting objects, $IN_t = \{IN_{t-1}, IN_{t-2}, \ldots, IN_{t-n}\}$, get the Euclidean distance between o_j and IN_{t-i}, $dist(o_j, IN_{t-i})$. Hence, if there are y objects of candidate skylines and p types of interesting objects with each type having approximately n objects, then the number of distances that needs to be measured $\approx y \times p \times n$. Table 6 shows the distances measured for each $o_j \in C - Sky_{G_a}$ and each IN. Here, there are $12 \times 9 = 108$ distances that are captured.

(b) For each $o_j \in C - Sky_{G_i}$ and for each set of t type of interesting objects, $IN_t = \{IN_{t-1}, IN_{t-2}, \ldots, IN_{t-n}\}$, the closest interesting object to the object o_j is determined based on *Definition* 11 *Closest Property of a t Type of Interesting Objects*. The distance of the closest interesting object, say IN_{t-i}, to o_j is then captured. This will produce an object o_j with p values of distances denoted as $o_j\left(cd_1, cd_2, \ldots, cd_p\right)$ with cd_h represents the closest distance of an interesting object of type h to the object o_j. Here, each p type can be regard as a dimension while cd_p is a value of the dimension p. By applying *Definition* 11 *Closest Property of a t Type of Interesting Objects* to the example given in Table 6, the closest object of each type to each $C - Sky_{G_a}$ is as given in Table 7. Meanwhile, Table 8 presents the closest distance values of each type to each $C - Sky_{G_a}$. For instance, the closest interesting objects of types 1, 2, and 3 to o_2 are IN_{1-1}, IN_{2-2}, and IN_{3-3}, with distances 7.61, 5, and 6.5, respectively.

(c) The final skylines are determined by performing the conventional skyline algorithm over the candidate skylines with p types of interesting objects as dimensions and distances as the values used in the pairwise comparisons. Here, *Definition* 15 *Skylines of a Space* is applied. For instance, $o_6(3, 3.16, 2.06)$ is said to dominate $o_2(7.61, 5, 6.5)$ since $\forall t \in \{1, 2, 3\}$, $o_6.t < o_2.t$. However, $o_9(2.23, 4.92, 3)$ and $o_{18}(6.08, 6.72, 2)$ are

TABLE 7. The closest object of each type to each $C - Sky_{Gt}$.

ID	IN_{1-1}	IN_{1-2}	IN_{1-3}	IN_{2-1}	IN_{2-2}	IN_{2-3}	IN_{3-1}	IN_{3-2}	IN_{3-3}
o_2	7.61	-	-	-	5	-	-	-	6.5
o_5	4.47	-	-	-	2.23	-	-	-	3.35
o_6	3	-	-	-	3.16	-	-	-	2.06
o_7	1.41	-	-	-	4.12	-	-	-	1.5
o_8	3.36	-	-	5.54	-	-	-	-	3.35
o_9	-	2.23	-	4.92	-	-	-	3	-
o_{18}	6.08	-	-	-	-	6.72	2	-	-
o_{20}	2.87	-	-	-	6.44	-	-	-	3.77
o_{25}	-	2.23	-	6.02	-	-	-	3.60	-
o_{26}	10	-	-	-	-	12.81	5	-	-
o_{32}	-	10	-	13.82	-	-	8.54	-	-
o_{40}	-	5.39	-	8.76	-	-	6.44	-	-

said not to dominate each other as o_9 is better than o_{18} in type 1 and type 2 and worse than o_{18} in type 3. Consequently, o_5 dominates o_2, o_7 dominates $o_8, o_9, o_{18}, o_{20}, o_{25}, o_{26}, o_{32}$, an o_{40}; while o_6 is not dominated by any other objects. As a result, the final skylines, $Sky_{G_a} = \{o_5, o_6, o_7\}$.

VI. RESULTS AND DISCUSSION

A. EXPERIMENTAL SETTINGS

To fairly evaluate the performance and prove the efficiency of *RSGU* and *ERSGU*, several extensive experiments are designed. These experiments are conducted on Intel Core i7 3.6GHz processor with 32GB of RAM and Windows 8 professional. The implementation of *RSGU* and *ERSGU* was done on VB.NET 2013. The performance results of *RSGU* and *ERSGU* are compared to the *VR* algorithm proposed by [63]. To the best of our knowledge, the works that are closely related to our work are by [60], [61], [62], and [63]; with [63] being the most recent among the list above.

In validating the correctness of *RSGU*, the following is conducted: The *VR* algorithm and the *RSGU* framework were run over a given data set and a group of users, G_a, to derive a set of skylines, $Sky_{G_a} - VR$ and $Sky_{G_a} - RSGU$, respectively. Then, given another group of users, G_b, the *VR* algorithm and the *RSGU* framework were run again and the set of skylines produced, namely: $Sky_{G_b} - VR$ and $Sky_{G_b} - RSGU$ are recorded. Intuitively, the correctness of *RSGU* is proven as the skyline objects produced by the *VR* algorithm, $Sky_{G_a} - VR$, is equal to the skyline objects produced by the *RSGU* framework, i.e. $Sky_{G_a} - VR = Sky_{G_a} - RSGU$. Similarly, $Sky_{G_b} - VR = Sky_{G_b} - RSGU$.

Meanwhile, the correctness of *ERSGU* framework could not be verified since both the *VR* algorithm and the *RSGU* framework that utilised the same evaluation criteria do not consider the closeness of the objects to other interesting objects. Undoubtedly, the skyline objects produced by the *ERSGU* framework are different from those produced by the *VR* algorithm and the *RSGU* framework.

Two types of data sets are used in the experiments, namely: *synthetic* and *TIGER* data sets. The *synthetic* data set is used to simulate several experimental settings to reflect all possible real settings. The *synthetic* data set is generated in such a way that all objects are independent with uniform distribution.

TABLE 8. The closest distance value of type 1, 2, and 3 to each $C - Sky_{Ga}$.

ID	IN_1	IN_2	IN_3
o_2	7.61	5	6.5
o_5	4.47	2.23	3.35
o_6	3	3.16	2.06
o_7	1.41	4.12	1.5
o_8	3.36	5.54	3.35
o_9	2.23	4.92	3
o_{18}	6.08	6.72	2
o_{20}	2.87	6.44	3.77
o_{25}	2.23	6.02	3.60
o_{26}	10	12.81	5
o_{32}	10	13.82	8.54
o_{40}	5.39	8.76	6.44

This is in line with the settings used in previous works [60], [61], [63]. In addition, in skyline queries, the *synthetic* data set is commonly used by previous researchers in evaluating the performance of their proposed approaches [70], [4], [81], [72], [73], [54], [51], [55], [59], [52]. On the other hand, the *TIGER* data set is a real data set from the line segment data of Long Beach. It is used by previous works that are related to spatial skyline queries [60], [61], [63]. The set of objects is prepared by extracting the midpoint of each road line segment. The data set contains 50,747 objects. There are 8 different types of objects, namely: *hospital*, *restaurant*, *church*, *school*, *institution*, *building*, *hotel*, and *populated place*. Each type of object has a different number of objects, a different list of non-spatial attributes, and spatial attributes. Skyline computation requires objects to be of same arity while the domination test is performed only on numerical values. Since *rate* and *price* are the only dimensions with numerical values, thus these dimensions are used as the evaluation criteria of the skyline computation.

Each experiment is run 10 times and the average value of these runs is reported. In deriving the set of skylines, it is assumed that lower values are preferable compared to higher ones. The performance measurement used in the experiments is processing time as it is the most commonly used measurement in evaluating the performance of skyline algorithm [63]. The processing time is evaluated for different parameter settings that are:

(i) number of users in a group – which is varied with minimum 4 users and maximum 25 users,
(ii) number of groups of users – which is varied with minimum 2 groups and maximum 32 groups,
(iii) overlapping region – the overlapped regions between different groups of users are controlled with 20%, 40%, 60%, 80%, and 100% of overlapped,
(iv) space size – the space area of the *synthetic* and *TIGER* data sets is as presented in Table 9,
(v) density – for the *TIGER* data set, the density of the objects is set to 0.56%, 1.60%, 7%, 15%, and 34%,
(vi) dimensionality – for both the *TIGER* and *synthetic* data sets, the number of dimensions is varied with 2, 4, 6, 8, and 10 dimension, and
(vii) number of objects – the initial size of the Long Beach from the *TIGER* data set is 50,747 objects while the

TABLE 9. The parameter settings of the *synthetic* and *TIGER* data sets.

Parameter Settings	Data Sets	
	Synthetic	Long Beach Tiger
Number of dimensions	2, 4, **6**, 8, 10	**2**, 4, 6, 8, 10
Number of Groups of Users	2, 4, 8, **16**, 32	2, 4, 8, **16**, 32
Number of Users in Groups	4, 8, **15**, 20, 25	4, 8, **15**, 20, 25
Percentage of Overlapping Region	20%, **40%**, 60%, 80%, 100%	20%, **40%**, 60%, 80%, 100%
Number of Objects	20,000, **50,000**, 80,000	**50,747**
Space	[0, 250]*[0, 250], [0, 500]*[0, 500], [0, 750]*[0, 750], **[0, 1000]*[0, 1000]**	[0, 250]*[0, 250], [0, 500]*[0, 500], [0, 750]*[0, 750], **[0, 1000]*[0, 1000]**
Density	-	0.56%, 1.60%, 7%, 15%, 34%

number of objects of the *synthetic* data set is varied with 50K as the minimum and 80K as the maximum number of objects.

Meanwhile, the location of each user is randomly generated within a given space size. These parameter settings are clearly shown in Table 9 with values in bold representing the default values.

B. THE EXPERIMENTAL RESULTS

This section presents the experimental results of the *RSGU* and *ERSGU* frameworks that are designed with the aim at deriving skyline objects of a group of users in a predetermined region. The performance of both frameworks is measured with regard to processing time with different parameter settings as discussed in Part *A* and presented in Table 8. These results are compared to the results of *VR* [63], based on the *synthetic* and real data sets. Both the *RSGU* and *ERSGU* frameworks derived skyline objects by analysing the locations of the users, i.e. spatial attributes, as well as the spatial and non-spatial attributes of the set of objects that is within a predetermined region of the group of users. However, in the *ERSGU* framework, the closeness of an object to the desirable facilities or other interesting objects in the region is also considered. Both frameworks work by partitioning the region into smaller units called fragments in such a way that overlapping areas between the current and previous visited regions can be easily determined. The results of computing the skylines of each fragment, known as fragment skylines, are saved and utilised in the skyline computation of subsequent requests. Meanwhile, the *VR* algorithm is performed repeatedly for each group of user's query without exploiting the previous skyline computation results.

Effect of Number of Groups of Users — The number of groups of users is one of the factors that has significant effect on the performance of skyline algorithms in processing skyline queries of a group of users. In this section, the experimental results of the proposed solutions, *RSGU* and *ERSGU*, and the previous algorithm, namely: *VR* [63] are illustrated, for both the *synthetic* and *TIGER* data sets with respect to processing time, by varying the number of groups of users from 2 – 32 as applied in [63]. The parameter settings for the *synthetic* data set are as follows: the number of dimensions is set to 6, the number of users in a group is set to 15 in a fixed space [0, 1000]*[0, 1000] with 40% overlapping region, while the number of objects is set to 50K. Meanwhile, the parameter settings for the *TIGER* data set are as follows: the number of dimensions is maintained to 2, the number of users in a group is set to 15 in a fixed space [0, 1000]*[0, 1000] with 40% overlapping region, while the number of objects is 50,747.

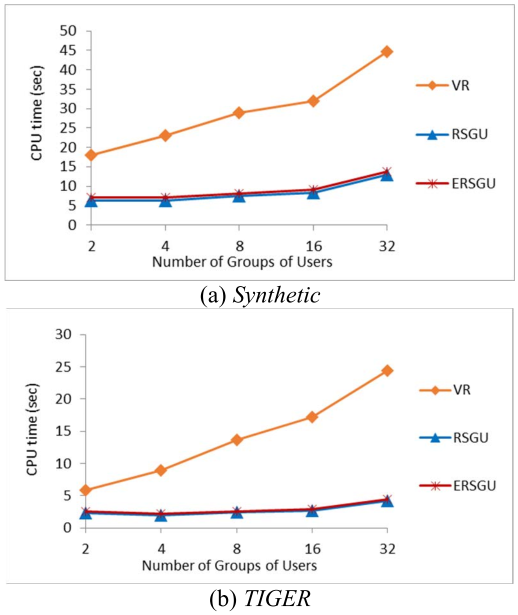


FIGURE 14. The results of processing time with varying number of groups of users.

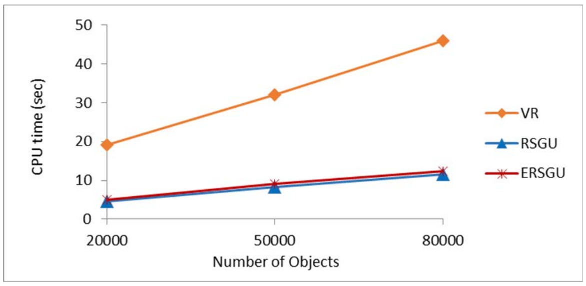
Fig. 14(a) and 14(b) present the processing time achieved by the *RSGU*, *ERSGU*, and *VR* algorithm [63] based on the *synthetic* and *TIGER* data sets, respectively, with the number of groups sets from 2 to 32 groups. The processing time is calculated based on the following formula:

$$\sum\nolimits_{i=1}^{n} processing\ time_i \qquad (3)$$

where n is the number of groups in a run. For instance, if the number of groups is 4, the processing time is calculated as $\sum\nolimits_{i=1}^{4} processing\ time_i$. The *VR* algorithm is performed repeatedly for each group of user's query in which the predetermined region of a group is explored even though it has been analysed during the skyline computation of the earlier groups. Meanwhile, for both *RSGU* and *ERSGU*, only the fragment skylines that are related to the identified overlapping area need to be analysed.

Intuitively, when the number of groups increases, the processing time also increases which can be clearly seen through the performance of the *RSGU*, *ERSGU*, and *VR* algorithm. Nonetheless, both *RSGU* and *ERSGU* show a steady performance for all runs with lesser processing time as compared to the *VR* algorithm. The processing time of *ERSGU* is slightly higher than *RSGU* since it has an additional evaluation criterion to be analysed, i.e. the closeness of an object to the desirable facilities or other interesting objects in the region. Similar trends as presented in Fig. 14(a) can be seen in Fig. 14(b). On the average, *RSGU* and *ERSGU* gained 70% and 67% improvements for the *synthetic* data set, respectively, and 72% and 69% for the *TIGER* data set, respectively; compared to the *VR* algorithm.

Effect of Number of Objects – In this study, the effect of number of objects on the performance of *RSGU*, *ERSGU*, and *VR* algorithm [63] is investigated. It is one of the important factors that has high impact on the skyline algorithms in deriving skyline objects. The parameter settings for the *synthetic* data set are as follows: the number of dimensions is fixed to 6, the number of groups is set to 16 groups with each group consisting of 15 users, the overlapping region is fixed to 40%, and the number of objects is varied with the following values: 20K, 50K, and 80K. Since the *TIGER* data set contains only 50,747 objects, hence it is excluded from this experiment.



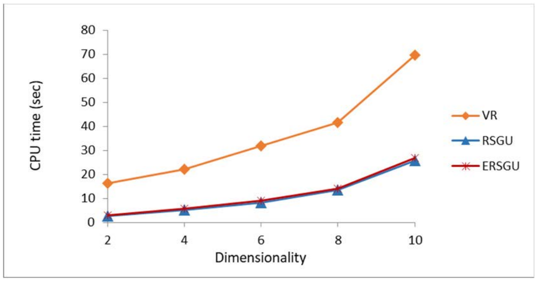
Synthetic

FIGURE 15. The results of processing time with varying number of objects.

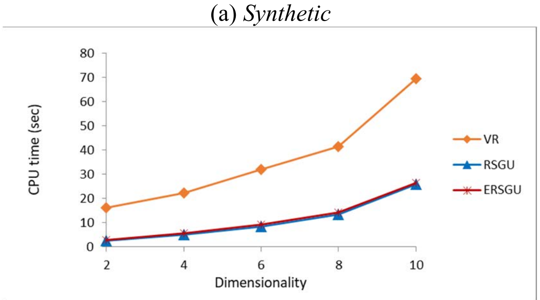
Fig. 15 presents the processing time achieved by the *RSGU*, *ERSGU*, and *VR* algorithm [63] based on the *synthetic* data set, with the number of objects sets as 20K, 50K, and 80K. The processing time is calculated based on formula (3) with the number of groups, $n = 16$. The *VR* algorithm is performed repeatedly for each group of user's query in which the predetermined region of a group is explored repeatedly even though it has been analysed during the skyline computation of the earlier groups. Meanwhile, for both *RSGU* and *ERSGU*, only the fragment skylines that are related to the identified overlapping area need to be analysed.

Obviously, when the number of objects increases, the processing time also increases which can be clearly seen through the performance of the *RSGU*, *ERSGU*, and *VR* algorithm. Nonetheless, both *RSGU* and *ERSGU* show a steady performance for all runs with lesser processing time as compared to the *VR* algorithm. The processing time of *ERSGU* is slightly higher than *RSGU* since it has an additional evaluation criterion to be analysed, i.e. the closeness of an object to the desirable facilities or other interesting objects in the region. On the average, *RSGU* and *ERSGU* gained 74% and 72%, improvements, respectively, compared to the *VR* algorithm for the *synthetic* data set.

Effect of Data Dimensionality – Besides the number of groups of users and the number of objects, data dimensionality is also one of the factors that has substantial effect on the performance of skyline algorithms in identifying the skyline objects of a group of users. In this section, the experimental results of the proposed solutions, *RSGU* and *ERSGU*, and the previous algorithm, namely: *VR* [63] are illustrated, for both the *synthetic* and *TIGER* data sets with respect to processing time, by varying the number of dimensions from 2 – 10 dimensions. The parameter settings for the *synthetic* data set are as follows: the number of objects is fixed to 50K, the number of groups is set to 16 groups with each group consisting of 15 users, and the overlapping region is fixed to 40%. For the *TIGER* data set, the same parameter settings as above are used except that the number objects is maintained to its initial number, i.e. 50,747 objects.



(a) *Synthetic*



(b) *TIGER*

FIGURE 16. The results of processing time with varying dimensionality.

Fig. 16(a) and 16(b) present the processing time achieved by the *RSGU*, *ERSGU*, and *VR* algorithm [63] based on the *synthetic* and *TIGER* data sets, respectively, with number of dimensions varied from 2 – 10 dimensions. The processing time is calculated based on the formula (3) with the number of groups, $n = 16$. Obviously, when the number of dimensions increases, the processing time also increases which can be clearly seen through the performance of the *RSGU*, *ERSGU*, and *VR* algorithm. Nonetheless, both *RSGU* and *ERSGU* show a steady performance for all runs with lesser processing time as compared to the *VR* algorithm. The processing time of *ERSGU* is slightly higher than *RSGU* since it has an additional evaluation criterion to be analysed, i.e. the closeness of an object to the desirable facilities or other interesting objects in the region. On the average, *RSGU* and *ERSGU* gained 69% and 67% improvements, respectively, for the *synthetic* data

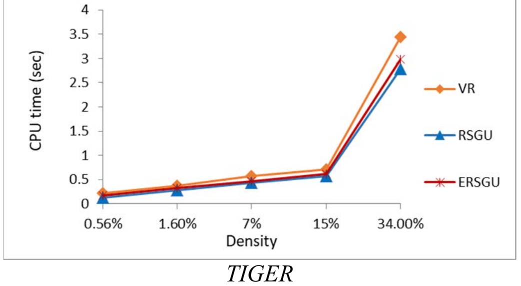
TABLE 10. The density rate of the types of objects in the *TIGER* data set.

Types of objects	% of the type of object in the whole population	No. of objects
Hospital	0.56	284
Restaurant	1.60	812
Church	7.00	3,552
School	15.00	7,612
Institution	34.00	17,254
Building	10.84	5,502
Hotel	13.00	6,597
Populated place	18.00	9,134

set and 69% and 67% improvements, respectively, for the *TIGER* data set compared to the *VR* algorithm.

Effect of Density – In this study, the effect of density on the performance of *RSGU*, *ERSGU*, and *VR* [63] is investigated. For this experiment, only the *TIGER* data set is considered. The *TIGER* data set consists of eight types of objects, namely: *hospital*, *restaurant*, *church*, *school*, *institution*, *building*, *hotel*, and *populated place*. This is represented in Table 10. The *% of the type of object in the whole population* and *No. of objects* reflect the density rate of a particular type of object in the area. For instance, the number of hospitals is 284 which is 0.56% of the whole population, i.e. 0.56% × 50747. Meanwhile, *institution* is the densest objects with density rate = 34%. Intuitively, the higher the density rate; the higher is the processing time as more objects need to be analysed.

In this section, the experimental results of the proposed solutions, *RSGU* and *ERSGU*, and the previous algorithm, namely: *VR* [63] are illustrated, for the *TIGER* data set with respect to processing time, with various density rates as follows: 0.56% (*hospital*), 1.60% (*restaurant*), 7.00% (*church*), 15.00% (*school*), and 34.00% (*institution*). These density rates reflect the least dense to the densest that are available in the data set. The parameter settings used are as follows: the number of objects is 50,747, the number of groups is set to 16 groups with each group consisting of 15 users, the number of dimensions and the overlapping region is fixed to 2 and 40%, respectively.

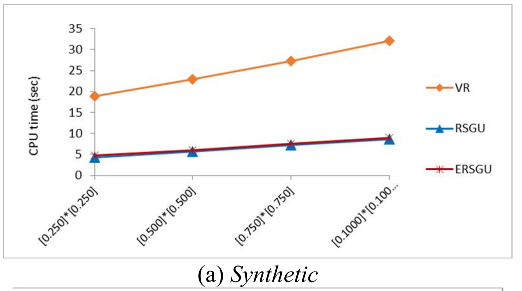


TIGER

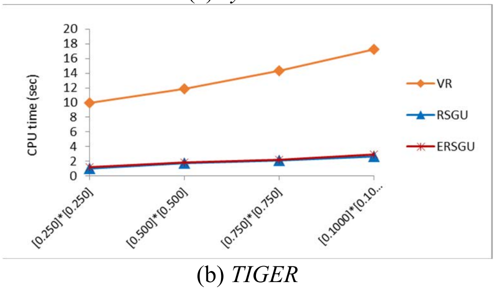
FIGURE 17. The results of processing time with varying density rate.

Fig. 17 shows the performance of *RSGU*, *ERSGU*, and *VR* algorithm [63] with regard to processing time. The performance of *RSGU*, *ERSGU*, and *VR* algorithm shows similar trends in which it starts to show a drastic increment when the density rate is 15.00% until it reaches to 34.00%. This is due to the fact that the number of *institutions* (34.00%) is slightly more than twice the number of *schools* (15.00%). Despite that, both *RSGU* and *ERSGU* show a better performance for all runs with lesser processing time as compared to the *VR* algorithm. The percentage of improvement gained by *RSGU* and *ERSGU* is 21% and 14%, respectively. The processing time of *ERSGU* is slightly higher than *RSGU* since it has an additional evaluation criterion to be analysed, i.e. the closeness of an object to the desirable facilities or other interesting objects in the region. For instance, if the object of interest is *restaurant*, then the desirable facilities or other interesting objects are *hospital*, *church*, *school*, and *institution*.

Effect of Space Size – In this study, the effect of space size on the performance of *RSGU*, *ERSGU*, and *VR* algorithm is also investigated. In this section, the experimental results of the proposed solutions, *RSGU* and *ERSGU*, and the previous algorithm, namely: *VR* [63] are illustrated for both the *synthetic* and *TIGER* data sets with respect to processing time, by varying the space size as follows: [0, 250]*[0, 250], [0, 500]*[0, 500], [0, 750]*[0, 750], and [0, 1000]*[0, 1000]. The parameter settings used for the *TIGER* data set are as follows: the number of objects is fixed to 50,747 objects, the number of groups is set to 16 groups with each group consisting of 15 users, and the overlapping region is set to 40%. For the *synthetic* data set, the same parameter settings as above are used except that the number objects is set to 50K objects.



(a) *Synthetic*

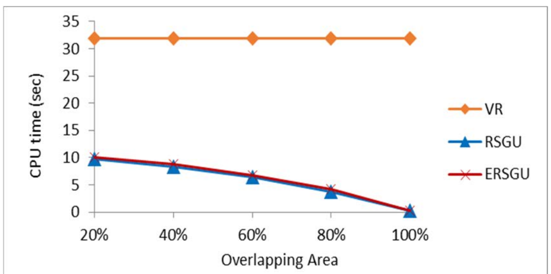


(b) *TIGER*

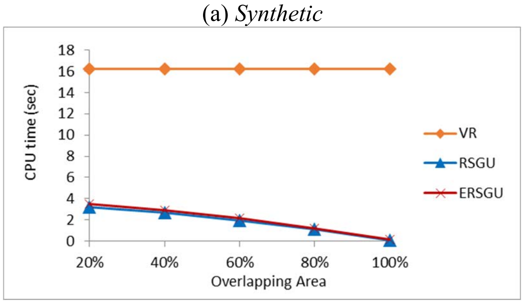
FIGURE 18. The results of processing time with varying space size.

Fig. 18(a) and 18(b) present the processing time achieved by the *RSGU*, *ERSGU*, and *VR* algorithm [63] based on the *synthetic* and *TIGER* data sets, respectively, with different space sizes. The processing time is calculated based on the formula (3) with the number of groups, $n = 16$. Obviously, the bigger the space size, the more objects it covered; hence the higher is the processing time. The *VR* algorithm is performed repeatedly for each group of user's query in which the predetermined region of a group is explored repeatedly even though it has been analysed during the skyline computation of the earlier groups of users. Meanwhile, for both *RSGU* and *ERSGU*, only the fragment skylines that are related to the identified overlapping area need to be analysed. From the figure, both *RSGU* and *ERSGU* show a steady performance with a slight increment in each iteration. Based on this analysis, *RSGU* and *ERSGU* gained 73% and 74% improvements, respectively, for the *synthetic* data set, and 82% and 83% improvements, respectively, for the *TIGER* data set, compared to the *VR* algorithm.

Effect of Overlapping Region – Another factor that has a significant effect on the performance of skyline algorithms for a group of users is the overlapping region covered between the groups of users. In this section, the experimental results of the proposed solutions, *RSGU* and *ERSGU*, and the previous algorithm, namely: *VR* [63] are illustrated, for both the *synthetic* and *TIGER* data sets with respect to processing time, by varying the percentage of overlapping region from 20% – 100%. The parameter settings for the *synthetic* data set are as follows: the number of objects is fixed to 50K, the number of groups is set to 16 groups with each group consisting of 15 users, and the number of dimensions is fixed to 6. For the *TIGER* data set, the same parameter settings as above are used except that the number objects is maintained to its initial number, i.e. 50,747 objects in [0,1000]*[0,1000], and each object is with 2 dimensions.



(a) *Synthetic*



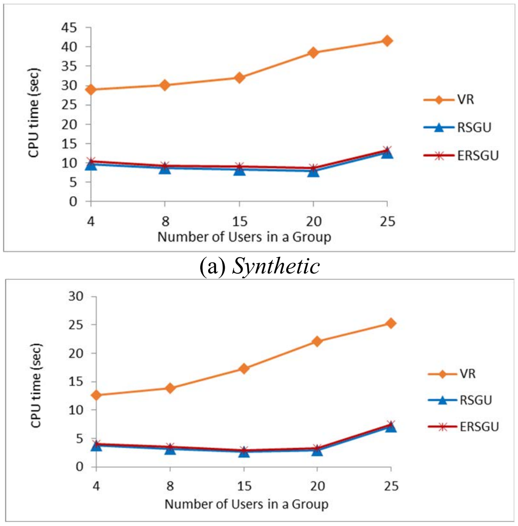
(b) *TIGER*

FIGURE 19. The results of processing time with varying percentage of overlapping area.

Fig. 19(a) and 19(b) present the processing time achieved by the *RSGU*, *ERSGU*, and *VR* [63], based on the *synthetic* and *TIGER* data sets, respectively, with varying percentage of overlapping region. The processing time is calculated based on the formula (3) with the number of groups, $n = 16$. From these figures, both the *RSGU* and *ERSGU* show a steady performance which reflects that when the percentage of overlapping region increases, the processing time decreases. This is due to the fact that both the *RSGU* and *ERSGU* exploit the skyline computation results of the previous groups of users that are associated to the identified overlapping region. Hence, the higher the percentage of overlapping region means the higher the percentage of the area that has been explored in the earlier skyline computations of the groups of users. In both solutions, *RSGU* and *ERSGU*, rescanning of the overlapping region and recomputation of skyline objects within the overlapping region are avoided. Interestingly, when the percentage of overlapping region is 100%, which implies that the region covered by the current group of users is the exact same region that has been explored in the skyline computations of the previous groups of users; shows the processing time taken is almost 0 because of the skyline results are the same. However, the *VR* algorithm shows a steady performance for all runs which clearly indicates that the overlapping region has no significant effect on the performance of the *VR* algorithm. This is true since the *VR* algorithm is performed repeatedly for each group of user's query in which the predetermined region of a group is explored repeatedly even though it has been analysed during the skyline computation of the earlier groups. Based on this analysis, *RSGU* and *ERSGU* gained 82% and 81% improvements, respectively, for the *synthetic* data set, and 87% and 86% improvements, respectively, for the *TIGER* data set, compared to the *VR* algorithm.

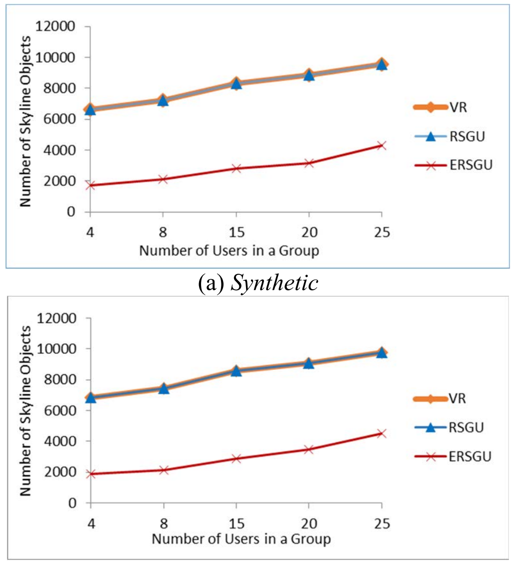
Effect of Number of Users in a Group – Another factor that has a major impact on the performance of skyline algorithms in processing the skyline queries of a group of users is the number of users in a group. In this section, the experimental results of the proposed solutions, *RSGU* and *ERSGU*, and the previous algorithm, namely: *VR* [63] are illustrated, for both the *synthetic* and *TIGER* data sets with respect to processing time by varying the number of users in a group from 4 – 25 as applied in the previous study [63]. The parameter settings for the *synthetic* data set are as follows: the number of objects is fixed to 50K, the number of groups is set to 16 groups in a fixed space [0, 1000]*[0, 1000] with 40% overlapping region, while the number of dimensions is fixed to 6. For the *TIGER* data set, the same parameter settings as above are used except that the number objects is maintained to its initial number, i.e. 50,747 objects, with the number of dimensions fixed to 2.

Fig. 20(a) and 20(b) present the processing time achieved by *RSGU*, *ERSGU*, and *VR* [63] based on the *synthetic* and *TIGER* data sets, respectively, with the number of users in a group sets to 4, 8, 15, 20, and 25. The processing time is calculated based on the formula (3) with the number of groups, $n = 16$. From these figures, both *RSGU* and *ERSGU* show similar performance for all runs with lesser processing time as compared to the *VR* algorithm. The processing time of *ERSGU* is slightly higher than *RSGU* since it has an additional evaluation criterion to be analysed, i.e. the closeness of an object to the desirable facilities or other interesting objects in the region. On the average, *RSGU* and *ERSGU* gained 72%



(b) *TIGER*

FIGURE 20. The results of processing time with varying number of users in a group.



(b) *TIGER*

FIGURE 21. The results of number of skyline objects with varying number of users in a group.

and 70% improvements, respectively, for the *synthetic* data set and 76% and 78%, respectively, for the *TIGER* data set, compared to the *VR* algorithm.

Number of Skyline Objects – Since the number of skyline objects is a significant factor in validating the correctness of the skyline algorithms in processing skyline queries, thus investigating the performance of *RSGU*, *ERSGU*, and the previous algorithm, *VR* [63] with regard to the skyline objects derived by these solutions is inevitable. In this section, the experimental results of the proposed solutions, *RSGU* and *ERSGU*, and the previous algorithm, namely: *VR* [63], for both the *synthetic* and *TIGER* data sets with respect to the number of skyline objects derived by these solutions are illustrated. The parameter settings for the *synthetic* data set are as follows: the number of objects is fixed to 50K, the number of groups is set to 16 groups in a fixed space [0, 1000]*[0, 1000] with 40% overlapping region, the number of dimensions is fixed to 6, while the number of users in a group is varied from 4 – 25. For the *TIGER* data set, the same parameter settings as above are used except that the number objects is maintained to its initial number, i.e. 50,747 objects, with the number of dimensions fixed to 2.

Fig. 21(a) and 21(b) present the number of skyline objects derived by the *RSGU*, *ERSGU*, and the *VR* algorithm [63], based on the *synthetic* and *TIGER* data sets, respectively. From these figures, it is obvious that the number of skyline objects derived by *RSGU* and *VR* is the same for all runs which verified the correctness of *RSGU*. This is because in deriving the skyline objects, both *RSGU* and *VR* utilised the same evaluation criteria, namely: spatial and non-spatial attributes of the objects. However, as expected the number of skyline objects produced by *ERSGU* is different from the number of skyline objects obtained by both the *RSGU* and *VR* algorithm since *ERSGU* has an additional evaluation criterion to be analysed, i.e. the closeness of an object to the desirable facilities or other interesting objects in the region.

C. TIME COMPLEXITY ANALYSIS

This section presents the time complexity analysis of the proposed frameworks, *Region-based Skyline for a Group of Users* (*RSGU*) and *Extended Region-based Skyline for a Group of Users (ERSGU)*. Since *ERGSU* utilises a different set of evaluation criteria in deriving the final skyline objects, hence we only report its time complexity without comparing it to the baseline method. On the other hand, the time complexity of *RSGU* is compared to a baseline method (*BM*) which utilises the conventional skyline algorithm in deriving the skyline objects for a group of users. In this method, each group of users is treated separately. This approach is assumed by many methods including the *VR* algorithm [63].

The time complexity analysis is based on the following: Given a data set $D = < R, U, O >$, where $U = \{u_1, u_2, \ldots, u_n\}$ is a list of n users and $O = \{o_1, o_2, \ldots, o_m\}$ is a list of m objects. Let $G_p = \{u_1, u_2, \ldots, u_p\}$ be a group of p users where $G_p \subset U$ in region R_p. Assume that $O_r = \{o_1, o_2, \ldots, o_r\}$ is the list of objects that is within the region R_p. Table 11 and Table 12 present the time complexity at each step of the *RSGU* and *ERSGU*, respectively.

In order to compare the time complexity of *RSGU* against the baseline method (*BM*), assume the following:

Given a group of users, $G_p = \{u_1, u_2, \ldots, u_p\}$, where $G_p \subset U$, and the candidate skylines of G_p in region R_p denoted as CS_{G_p} with c cardinality. Find the skylines of a group of users $G_q = \{u_1, u_2, \ldots, u_q\}$ in region R_q, i.e. CS_{G_q}, where $G_q \subset U$, $G_q \neq G_p$, and $R_q \cap R_p \neq \emptyset$. Assume that $O_r = \{o_1, o_2, \ldots, o_r\}$ is the list of objects that is within the region R_p; $O_s = \{o_1, o_2, \ldots, o_s\}$ is the list of objects that is within the region R_q, u is the number of candidate skylines of CS_{G_p} that are in the overlapping area O_R, and v is the number of objects that are in the non-overlapping area $\neg O_R$. Obviously, $u \leq r$, $u \leq s$, $u \leq c$, and $u + v < s$.

TABLE 11. **The time complexity of *RSGU*.**

Steps	Time Complexity	Remarks
(1) Identify the centroid	$O(1)$	The centroid is identified using Equation (1).
(2) Construct a search region	$O(r) + O(p) + O(1)$	1. Identify the closest object to the centroid: r iterations, where r is the number of objects within the region R_p. 2. Construct a search region of each user in the G_p: p iterations, where p is the number of users in G_p. 3. Construct a search region for G_p: 1 iteration, by performing union over the search region of each user.
(3) Identify the overlapping region	$O(1) + O(\log M\ n)$	1. Identify the overlapping region based on the polygon's vertices: 1 iteration. 2. Traverse the R-tree to identify the fragments that are within the overlapping region: $O(\log M\ n)$, where M is the maximum number of entries and n is the minimum number of entries in a node.
(4) Construct the fragments of a search region	$O(2p \log 2p) + O(2p \log 2p) + O(k) + O(w^2)$	1. Sort the x-coordinates of each search region: $O(2p \log 2p)$ with 2 x-coordinates for each p user. 2. Sort the y-coordinates of each search region: $O(2p \log 2p)$ with 2 y-coordinates for each p user. 3. Construct the k fragments: k iterations. 4. Derive the non-spatial candidate skylines for k fragments: $\sum_{i=1}^{k} w_i(w_i - 1)/2$ iterations, where w_i is the number of objects of the i-th fragment; this is simplified as w^2 iterations.
(5) Construct the R-tree of the fragments	$O(\log M\ n)$	The R-tree is constructed based on the algorithms proposed by [30]: $O(\log M\ n)$, where M is the maximum number of entries and n is the minimum number of entries in a node.
(6) Derive the non-spatial skylines	$O(c^2)$	Apply the non-spatial dominance over the candidate skylines of the k fragments: $c(c-1)/2$ iterations, where c is the number of candidate skylines of the k fragments.
(7) Derive the spatial skylines	$O(pr) + O(r) + O(r \log r) + O(r^2)$	1. Calculate the distance between each user of G_p and each object of O_r: $p \times r$ iterations. 2. Calculate the $Sum\ Distance - o_i$ of each object of O_r: r iterations. 3. Sort the $Sum\ Distance - o_i$ of the O_r objects: $O(r \log r)$ 4. Apply the spatial dominance over the objects of O_r: $r(r-1)/2$ iterations.
(8) Derive the final skylines	$O(1)$	Union between the non-spatial skylines and spatial skylines.

TABLE 12. **The time complexity of *ERSGU*.**

Steps	Time Complexity	Remarks
Steps (1) till (7) are the same steps as *RSGU*.		
(8) Derive the candidate skylines	$O(1)$	Union between the non-spatial skylines and spatial skylines.
(9) Derive the final skylines	$O(yh) + O(yh) + O(y^2)$	1. Calculate the distance between each candidate skyline o_j and a set of p distinct type of interesting objects with h total number of objects: $y \times h$ iterations, where y is the number of candidate skylines. 2. Identify the closest interesting object of each type to the object o_j: $\sum_{i=1}^{p} y \times h_i$ iterations, where h_i is the number of objects of type p_i. 3. Perform the conventional skyline algorithm over the candidate skylines with the closest distance of each type as the evaluation criteria: $y(y-1)/2$ iterations.

The time complexity of both methods is analysed based on the following steps: (i) derive the final skylines of G_p and (ii) derive the final skylines of G_q. Here, we assumed that the region R_p is an unexplored region while $R_q \cap R_p \neq \emptyset$ implies that some parts of the region R_q have been analysed in step (i). To simplify the comparisons, only the steps followed by *BM* and *RSGU* with different time complexities are analysed.

(i) To derive the final skylines of G_p using the baseline method, similar steps to the steps 1, 2, 6, 7, and 8 are performed. The objects analysed in Step 6 are those objects that are within the region R_p; hence the number of objects analysed is r with $r(r-1)/2$ pairwise comparisons [52]. The time complexity to derive the final skylines of G_p with the baseline method, $T(BM_{G_p})$, is given below:

$$T(BM_{G_p}) = T\ (Step\ 6) \approx O\left(r^2\right)$$

Meanwhile, utilising the *RSGU*, steps 1, 2, 4, 5, 6, 7, and 8 are performed. The time complexity to derive the final skylines of G_p with *RSGU*, $T(RSGU_{G_p})$, is as given below:

$$T\left(RSGU_{G_p}\right) = T\ (Step\ 4) + T\ (Step\ 5) + T\ (Step\ 6) = [O(2p \log 2p) + O(2p \log 2p) + O(k) + O(w^2)] + O(\log Mn) + O(c^2) \approx O(w^2) + O(c^2)$$

Since $w + c < r$, thus $T\left(RSGU_{G_p}\right) < T(BM_{G_p})$.

(ii) To derive the final skylines of G_q using the baseline method, similar steps to the steps 1, 2, 6, 7, and 8 are performed. The objects analysed in Step 6 are those objects that are within the region R_q; hence the number of objects analysed is s with $s(s-1)/2$ pairwise comparisons [52]. The time complexity to derive the final skylines of G_q utilising the baseline method, $T(BM_{G_q})$ is as follows:

$$T(BM_{G_q}) = T\ (Step\ 6) \approx O\left(s^2\right)$$

Meanwhile, utilising the *RSGU*, steps 1, 2, 3, 4, 5, 6, 7, and 8 are performed.

$$T\left(RSGU_{G_q}\right) = T\ (Step\ 3) + T\ (Step\ 4) + T\ (Step\ 5) + T\ (Step\ 6) = O(\log Mn) + \left[O(2p \log 2p) + O(2p \log 2p) + O(k) + O(v^2)\right] + O(\log Mn) + O(u^2) \approx O\left(v^2\right) + O(u^2)$$

Step 4 is performed over the non-overlapping area involving v objects, while Step 6 is performed over the overlapping area in which u candidate skylines of CS_{G_p} are analysed.

Since $v + u < s$, thus $T\left(RSGU_{G_q}\right) < T(BM_{G_q})$.

(iii) The total time complexities for both methods are as follows:

$$T(BM) = T\left(BM_{G_p}\right) + T(BM_{G_q}) \approx O\left(r^2\right) + O\left(s^2\right)$$

$$T(RSGU) = T\left(RSGU_{G_p}\right) + T\left(RSGU_{G_q}\right) \approx O\left(w^2\right) + O\left(c^2\right) + \left(v^2\right) + O\left(u^2\right)$$

It is obvious that $T(RSGU) < T(BM)$ since $w + c < r$ and $v + u < s$.

VII. CONCLUSION

In this paper, we proposed the *Region-based Skyline for a Group of Users* (*RSGU*) and *Extended Region-based Skyline for a Group of Users (ERSGU)* frameworks that are designed to derive skyline objects which are point of interests (*PoIs*) to be recommended to a group of users. The skyline objects are derived by analysing both the locations of the users, i.e. spatial attributes, as well as the spatial and non-spatial attributes of objects that are within a predetermined region of the group of users. Two main aims have been set that are: (i) to avoid the process of rescanning the set of objects within a predetermined region that is known to have been previously visited by a group of users and (ii) to avoid the recomputation of skylines of a set of objects within a predetermined region that has been analysed in earlier computations of previously visited group of users. Meanwhile, *ERSGU* framework considers the closeness of the objects to other interesting objects in its solution as its additional feature. Several experiments have been conducted and the results show that both the *RSGU* and *ERSGU* outperform the work by [63] with respect to CPU time. There are several further enhancements that can be made based on the findings presented in the paper. These include (i) proposing an approach to continuously derive skyline objects by considering the movement of the users, (ii) incorporating the *Group-based skyline* in finding the optimal combinations of skyline objects [86] for a group of users, and (iii) embedding the proposed frameworks into a Travelling Recommender System similar to the work in [87].

ACKNOWLEDGMENT

All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] I. Ilyas, W. Aref, and A. Elmagarmid, "Supporting top-k join queries in relational databases," *VLDB J.*, vol. 13, no. 3, pp. 207–221, Sep. 2004.
[2] M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang, "Top-k query processing in uncertain databases," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 896–905.
[3] M. L. Yiu and N. Mamoulis, "Efficient processing of top-k dominating queries on multi-dimensional data," *Very Large Data Bases*, vol. 7, pp. 483–494, Sep. 2007.
[4] J. Lee, G.-W. You, and S.-W. Hwang, "Personalized top-k skyline queries in high-dimensional space," *Inf. Syst.*, vol. 34, no. 1, pp. 45–61, Mar. 2009.
[5] M. L. Yiu and N. Mamoulis, "Multi-dimensional top-k dominating queries," *VLDB J.*, vol. 18, no. 3, pp. 695–718, Jun. 2009.
[6] W. Son, F. Stehn, C. Knauer, and H.-K. Ahn, "Top-k Manhattan spatial skyline queries," *Inf. Process. Lett.*, vol. 123, pp. 27–35, Jul. 2017.
[7] R. Liu and D. Li, "Dynamic dimension indexing for efficient skyline maintenance on data streams," in *Proc. Conf. Database Syst. Adv. Appl.*, Sep. 2020, pp. 272–287.
[8] A. Alsaudi, Y. Altowim, S. Mehrotra, and Y. Yu, "TQEL: Framework for query-driven linking of top-k entities in social media blogs," in *Proc. Conf. Very Large Data Bases Endowment*, vol. 14, Jul. 2021, pp. 2642–2654.
[9] M. Costanzo, "Getting the best from skylines and top-k queries," 2022, *arXiv:2202.12618*.
[10] C. Zhang, P. Benz, A. Karjauv, J. W. Cho, K. Zhang, and I. S. Kweon, "Investigating top-k white-box and transferable black-box attack," in *Proc. Conf. IEEE/CVF Comput. Vis. Pattern Recognit.*, Mar. 2022, pp. 15085–15094.
[11] N. Ge, Z. Qin, P. Peng, M. Li, L. Zou, and K. Li, "A cost-driven top-K queries optimization approach on federated RDF systems," *IEEE Trans. Big Data*, early access, Mar. 3, 2022, doi: 10.1109/TBDATA.2022.3156090.
[12] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, Apr. 2001, pp. 421–430.
[13] K. L. Tan, P. K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," *Very Large Data Bases*, vol. 1, pp. 301–310, Sep. 2001.
[14] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proc. Conf. Very Large Data Bases*, Jan. 2002, pp. 275–286.
[15] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proc. Conf. Data Eng.*, Mar. 2003, pp. 717–719.

[16] P. Godfrey and W. Ning, "Relational preference queries via stable skyline," York Univ., Toronto, Toronto, ON, Canada, Tech. Rep. CS-2004-03, Aug. 2004.
[17] J. Pei, W. Jin, M. Ester, and Y. Tao, "Catching the best views of skyline: A semantic approach based on decisive subspaces," in *Proc. Conf. Very Large Data Bases*, Jan. 2005, pp. 253–264.
[18] X. Lin, Y. Yuan, W. Wang, and H. Lu, "Stabbing the sky: Efficient skyline computation over sliding windows," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, Apr. 2005, pp. 502–513.
[19] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting: Theory and optimizations," in *Proc. Conf. Intell. Inf. Process. Web Mining*, May 2005, pp. 595–604.
[20] Z. Huang and W. Wang, "Mining the useful skyline set based on the acceptable difference," in *Proc. Conf. Adv. Data Mining Appl.*, Aug. 2006, pp. 927–933.
[21] Z. Huang, H. Lu, B. C. Ooi, and A. K. H. Tung, "Continuous skyline queries for moving objects," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 12, pp. 1645–1658, Dec. 2006.
[22] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for uncertain data," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2010, pp. 1293–1296.
[23] B. Jiang and X. Liu, "Scaling of geographic space from the perspective of city and field blocks and using volunteered geographic information," *Geographical Inf. Sci.*, vol. 26, pp. 215–229, Feb. 2012.
[24] X. Ding, X. Lian, L. Chen, and H. Jin, "Continuous monitoring of skylines over uncertain data streams," *Inf. Sci.*, vol. 184, pp. 196–214, Feb. 2012.
[25] A. Arvanitis and G. Koutrika, "Towards preference-aware relational databases," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 426–437.
[26] C. Luo, Z. Jiang, W.-C. Hou, S. He, and Q. Zhu, "A sampling approach for skyline query cardinality estimation," *Knowl. Inf. Syst.*, vol. 32, no. 2, pp. 281–301, Aug. 2012.
[27] H. Choi, H. Jung, K. Y. Lee, and Y. D. Chung, "Skyline queries on keyword-matched data," *Inf. Sci.*, vol. 232, pp. 449–463, May 2013.
[28] S. Bothe, P. Karras, and A. Vlachou, "Eskyline: Processing skyline queries over encrypted data," in *Proc. Conf. Very Large Data Bases*, Aug. 2013, pp. 1338–1341.
[29] X. Lin, J. Xu, H. Hu, and W.-C. Lee, "Authenticating location-based skyline queries in arbitrary subspaces," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 6, pp. 1479–1493, Jun. 2014.
[30] W. Wang, J. Zhang, M.-T. Sun, and W.-S. Ku, "A scalable spatial skyline evaluation system utilizing parallel independent region groups," *VLDB J.*, vol. 28, no. 1, pp. 73–98, Feb. 2019.
[31] R. Liu and D. Li, "Efficient skyline computation in high-dimensionality domains," in *Proc. Conf. Extending Database Technol.*, Mar. 2020, pp. 459–462.
[32] R. J. Aarthi and B. Vinayagasundaram, "Dimensionally reduced optimal skyline evaluation by sampling in big data analytics," *Solid State Technol.*, vol. 64, pp. 1908–1927, Feb. 2021.
[33] A. Cuzzocrea, P. Karras, and A. Vlachou, "Effective and efficient skyline query processing over attribute-order-preserving-free encrypted data in cloud-enabled databases," *Future Gener. Comput. Syst.*, vol. 126, pp. 237–251, Jan. 2022.
[34] A. T. Kuo, H. Chen, L. Tang, W. S. Ku, and X. Qin, "ProbSky: Efficient computation of probabilistic skyline queries over distributed data," *IEEE Trans. Knowl. Data Eng.*, early access, Feb. 16, 2022, doi: 10.1109/TKDE.2022.3151740.
[35] S. S. Bavirthi and K. P. Supreethi, "An efficient framework for spatio-textual skyline querying and minimizing search space using R+ tree indexing technique," *Int. J. Inf. Technol.*, vol. 14, no. 3, pp. 1263–1271, May 2022.
[36] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2006, pp. 503–514.
[37] X. Miao, Y. Gao, G. Chen, B. Zheng, and H. Cui, "Processing incomplete K nearest neighbor search," *IEEE Trans. Fuzzy Syst.*, vol. 24, pp. 1349–1363, Jan. 2016.
[38] P. Srivastava and R. Khan, "A review paper on cloud computing," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 8, no. 6, pp. 17–20, 2018.
[39] D. Touliatou and M. A. Wheel, "K-dominance and size effect in mode i fracture of brittle materials with low to medium porosity," *Eng. Fract. Mech.*, vol. 201, pp. 269–281, Oct. 2018.
[40] T. Schibler, "Multidimensional team formation," Ph.D. dissertation, Dept. Comput. Sci., UC Santa Barbara, Santa Barbara, CA, USA, Sep. 2019.
[41] T. Schibler and S. Suri, "K-dominance in multidimensional data: Theory and applications," *Comput. Geometry*, vol. 87, Apr. 2020, Art. no. 101594.
[42] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos, "Continuous top-k dominating queries in subspaces," in *Proc. Panhellenic Conf. Informat.*, Aug. 2008, pp. 31–35.
[43] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos, "Continuous top-k dominating queries," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 840–853, May 2012.
[44] D. Amagata, T. Hara, and M. Onizuka, "Space filling approach for distributed processing of top-k dominating queries," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1150–1163, Jun. 2018.
[45] N. K. Rao, K. Radhika, D. Rahul, J. Thirupathi, and B. Madhuravani, "A survey on top-k dominating queries on any incomplete information," *Indian J. Pure Appl. Math.*, vol. 119, pp. 1807–1811, Mar. 2018.
[46] H. Zhu, X. Li, Q. Liu, and Z. Xu, "Top-k dominating queries on skyline groups," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 7, pp. 1431–1444, Jul. 2020.
[47] X. Miao, Y. Gao, S. Guo, L. Chen, J. Yin, and Q. Li, "Answering skyline queries over incomplete data with crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1360–1374, Apr. 2021.
[48] H. M. A. Fattah, K. M. A. Hasan, and T. Tsuji, "Weighted top-k dominating queries on highly incomplete data," *Inf. Syst.*, vol. 107, Jul. 2022, Art. no. 102008.
[49] C. Y. Chan, H. V. Jagadish, K. L. Tan, A. K. Tung, and Z. Zhang, "On high dimensional skylines," in *Proc. Conf. Extending Database Technol.*, Mar. 2006, pp. 478–495.
[50] J. Lee, H. Im, and G.-W. You, "Optimizing skyline queries over incomplete data," *Inf. Sci.*, vols. 361–362, pp. 14–28, Sep. 2016.
[51] A. Wahid and K. Kashyap, "Cassandra—A distributed database system: An overview," in *Emerging Technologies in Data Mining and Information Security*. Singapore: Springer, Dec. 2019, pp. 519–526.
[52] G. B. Dehaki, H. Ibrahim, F. Sidi, N. I. Udzir, A. A. Alwan, and Y. Gulzar, "Efficient computation of skyline queries over a dynamic and incomplete database," *IEEE Access*, vol. 8, pp. 141523–141546, 2020.
[53] G. B. Dehaki, H. Ibrahim, A. A. Alwan, F. Sidi, and N. I. Udzir, "Efficient skyline computation over an incomplete database with changing states and structures," *IEEE Access*, vol. 9, pp. 88699–88723, 2021.
[54] J. Liu, J. Yang, L. Xiong, J. Pei, and J. Luo, "Skyline diagram: Finding the Voronoi counterpart for skyline queries," in *Proc. Conf. Data Eng.*, Apr. 2018, pp. 653–664.
[55] K. Alami and S. Maabout, "A framework for multidimensional skyline queries over streaming data," *Data Knowl. Eng.*, vol. 127, May 2020, Art. no. 101792.
[56] V. Shepelev, A. Glushkov, Z. Almetova, and V. Mavrin, "A study of the travel time of intersections by vehicles using computer vision," in *Proc. 6th Int. Conf. Vehicle Technol. Intell. Transp. Syst.*, 2020, pp. 653–658.
[57] Y. Gulzar, A. A. Alwan, N. Salleh, and I. F. Al Shaikhli, "Processing skyline queries in incomplete database: Issues, challenges and future trends," *J. Comput. Sci.*, vol. 13, no. 11, pp. 647–658, Nov. 2017.
[58] Y. Gulzar, A. A. Alwan, H. Ibrahim, and Q. Xin, "D-SKY: A framework for processing skyline queries in a dynamic and incomplete database," in *Proc. 20th Int. Conf. Inf. Integr. Web-based Appl. Services*, Nov. 2018, pp. 164–172.
[59] M. B. Swidan, A. A. Alwan, S. Turaev, H. Ibrahim, A. Z. Abualkishik, and Y. Gulzar, "Skyline queries computation on crowdsourced-enabled incomplete database," *IEEE Access*, vol. 8, pp. 106660–106689, 2020.
[60] M. Sharifzadeh and C. Shahabi, "The spatial skyline queries," in *Proc. Conf. Very Large Data Bases*, Sep. 2006, pp. 751–762.
[61] M. Sharifzadeh, C. Shahabi, and L. Kazemi, "Processing spatial skyline queries in both vector spaces and spatial network databases," *ACM Trans. Database Syst.*, vol. 34, no. 3, pp. 1–45, Aug. 2009.
[62] M. Geng, M. S. Arefin, and Y. Morimoto, "A spatial skyline query for a group of users having different positions," in *Proc. 3rd Int. Conf. Netw. Comput.*, Dec. 2012, pp. 137–142.
[63] M. S. Arefin, G. Ma, and Y. Morimoto, "A spatial skyline query for a group of users," *J. Softw.*, vol. 9, no. 11, pp. 2938–2947, Nov. 2014.
[64] N. B. Moe, T. Dingsøyr, and K. Rolland, "To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in largescale agile software development," *Int. J. Inf. Syst. Project Manage.*, vol. 6, no. 3, pp. 45–59, Jan. 2022.
[65] V. Stray and N. B. Moe, "Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and slack," *J. Syst. Softw.*, vol. 170, Dec. 2020, Art. no. 110717.
[66] H. Cao, "Available technologies on improving the stability of polyphenols in food processing," *Food Frontiers*, vol. 2, no. 2, pp. 109–139, Jun. 2021.

[67] G. B. Dehaki, H. Ibrahim, N. I. Udzir, F. Sidi, and A. A. Alwan, "A fragmentation region-based skyline computation framework for a group of users," in *Proc. AI, Mach. Learn. Appl.*, Aug. 2021, pp. 35–50.

[68] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *Proc. Conf. Very Large Data Bases*, Aug. 2005, pp. 229–240.

[69] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2003, pp. 467–478.

[70] I. Bartolini, P. Ciaccia, and M. Patella, "SaLSa: Computing the skyline without scanning the whole sky," in *Proc. 15th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2006, pp. 405–414.

[71] M. A. M. Lawal, H. Ibrahim, N. F. M. Sani, and R. Yaakob, "An indexed non-probability skyline query processing framework for uncertain data," in *Proc. Conf. Adv. Mach. Learn. Technol. Appl.*, Feb. 2020, pp. 289–301.

[72] A. A. Alwan, H. Ibrahim, N. I. Udzir, and F. Sidi, "An efficient approach for processing skyline queries in incomplete multidimensional database," *Arabian J. Sci. Eng.*, vol. 41, no. 8, pp. 2927–2943, Aug. 2016.

[73] A. A. Alwan, H. Ibrahim, N. I. Udzir, and F. Sidi, "Processing skyline queries in incomplete distributed databases," *J. Intell. Inf. Syst.*, vol. 48, no. 2, pp. 399–420, Apr. 2017.

[74] Y. Gulzar, A. A. Alwan, and S. Turaev, "Optimizing skyline query processing in incomplete data," *IEEE Access*, vol. 7, pp. 178121–178138, 2019.

[75] N. H. M. Saad, H. Ibrahim, F. Sidi, R. Yaakob, and A. A. Alwan, "Efficient skyline computation on uncertain dimensions," *IEEE Access*, vol. 9, pp. 96975–96994, 2021.

[76] H. Wang and W. Zhang, "The τ-skyline for uncertain data," in *Proc. Conf. Can. Conf. Comput. Geometry*, Aug. 2014, pp. 326–331.

[77] N. H. M. Saad, H. Ibrahim, A. A. Alwan, F. Sidi, and R. Yaakob, "A framework for evaluating skyline query over uncertain autonomous databases," *Proc. Comput. Sci.*, vol. 29, pp. 1546–1556, Jan. 2014.

[78] N. H. M. Saad, H. Ibrahim, F. Sidi, R. Yaakob, and A. A. Alwan, "Computing range skyline query on uncertain dimension," in *Proc. Conf. Database Expert Syst. Appl.*, Sep. 2016, pp. 377–388.

[79] N. H. M. Saad, H. Ibrahim, F. Sidi, and R. Yaakob, "Non-index based skyline analysis on high dimensional data with uncertain dimensions," in *Proc. Conf. Baltic Conf. Databases Inf. Syst.*, Jul. 2018, pp. 272–286.

[80] N. H. M. Saad, H. Ibrahim, F. Sidi, and R. Yaakob, "Skyline probabilities with range query on uncertain dimensions," in *Proc. Conf. Adv. Comput. Commun. Comput. Sci.*, May 2019, pp. 225–242.

[81] M. Bai, J. Xin, and G. Wang, "Probabilistic reverse skyline query processing over uncertain data stream," in *Proc. Conf. Database Syst. Adv. Appl.*, Apr. 2012, pp. 17–32.

[82] Z. Dzolkhifli, H. Ibrahim, F. Sidi, L. S. Affendey, S. N. M. Rum, and A. A. Alwan, "A skyline query processing approach over interval uncertain data stream with k-means clustering technique," in *Proc. Conf. Adv. Databases, Knowl., Data Appl.*, Jun. 2019, pp. 51–56.

[83] G. B. Dehaki, H. Ibrahim, N. I. Udzir, F. Sidi, and A. A. Alwan, "A framework for processing skyline queries for a group of mobile users," in *Proc. 20th Int. Conf. Inf. Integr. Web-Based Appl. Services*, Nov. 2018, pp. 333–339.

[84] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. Conf. ACM SIGMOD Manage. Data*, Jun. 1984, pp. 47–57.

[85] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics*, 10th ed. New York, NY, USA: Wiley, Aug. 2013.

[86] W. Yu, J. Liu, J. Pei, L. Xiong, X. Chen, and Z. Qin, "Efficient contour computation of group-based skyline," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 7, pp. 1317–1332, Jul. 2020.

[87] R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, and X. Li, "Efficient user profiling based intelligent travel recommender system for individual and group of users," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 1018–1033, Jun. 2019.

GHONCHEH BABANEJAD DEHAKI was born in Tehran, Iran, in March 1984. She received the bachelor's degree in the field of software engineering from the Faculty of Computer Science, Iran University of Payame Noor, in 2009, and the master's degree in knowledge management with multimedia from Multimedia University (MMU), Malaysia, specializing in semantic web ontology and recommender systems. She is currently pursuing the Ph.D. degree in the field of database systems, specializing in region-based skyline computation for a group of users with the Universiti Putra Malaysia (UPM).

HAMIDAH IBRAHIM (Member, IEEE) received the Ph.D. degree in computer science from the University of Wales Cardiff, U.K., in 1998. She is currently a Full Professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her current research interests include databases (distributed, parallel, mobile, biomedical, and XML) focusing on issues related to integrity maintenance/checking, ontology/schema/data integration, ontology/schema/data mapping, cache management, access control, data security, transaction processing, query optimization, query reformulation, preference evaluation–context-aware, information extraction, concurrency control, and data management in mobile, grid, and cloud.

ALI A. ALWAN received the Master of Computer Science and Ph.D. degrees in computer science from the Universiti Putra Malaysia (UPM), Malaysia, in 2009 and 2013, respectively. He is currently an Assistant Professor at the School of Theoretical and Applied Science, Ramapo College of New Jersey, USA. His research interests include databases (incomplete, uncertain, and probabilistic), preference queries, web databases, query processing and optimization, data integration, location-based social networks (LBSN), recommendation systems, data mining, database in cloud, big data management, and crowd-sourced database.

FATIMAH SIDI (Member, IEEE) received the Ph.D. degree in management information system from the Universiti Putra Malaysia (UPM), Malaysia, in 2008. She is currently working as an Associate Professor in the discipline of computer science with the Department of Computer Science, Faculty of Computer Science and Information Technology, UPM. Her current research interests include knowledge and information management systems, data and knowledge engineering, database, and data warehouse.

NUR IZURA UDZIR received the Bachelor of Computer Science and Master of Science degrees from the Universiti Putra Malaysia (UPM), in 1996 and 1998, respectively, and the Ph.D. degree in computer science from the University of York, U.K., in 2006. She has been an Associate Professor at the Faculty of Computer Science and Information Technology, UPM, since 1998. Her research interests include access control, secure operating systems, intrusion detection systems, coordination models and languages, and distributed systems. She is a member of the IEEE Computer Society.

MA'ARUF MOHAMMED LAWAL received the bachelor's and master's degrees in computer science from Ahmadu Bello University, Zaria, Nigeria, and the Ph.D. degree from the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. He is currently a Senior Lecturer at the Department of Computer Science, Faculty of Physical Sciences, Ahmadu Bello University. His research interests include query optimization, data science, preference evaluation-context-aware, information extraction, concurrency control, database integration, data security, transaction processing, query reformulation, preference query evaluation, and knowledge-based representation.