IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 13, 2021, accepted June 6, 2021, date of publication June 17, 2021, date of current version June 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3090171

Efficient Skyline Computation Over an Incomplete Database With Changing States and Structures

GHAZALEH BABANEJAD DEHAKI[1], HAMIDAH IBRAHIM[1], (Member, IEEE), ALI A. ALWAN[2], FATIMAH SIDI[1], AND NUR IZURA UDZIR[1]

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia
[2]Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Kuala Lumpur 53100, Malaysia

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

This work was supported by the Ministry of Higher Education Malaysia under the Fundamental Research Grant Scheme (FRGS/1/2016/ICT04/UPM/01/2) and the Universiti Putra Malaysia.

ABSTRACT Skyline query has been studied extensively and a significant number of skyline algorithms have been proposed, mostly attempt to resolve the optimisation problem that is mainly associated with reduction in the processing time of skyline computations. While databases change their states and/or structures throughout their lifetime to reflect the current and latest information of the applications, the skyline set derived before changes are made towards the initial state of a database is no longer valid in the new state/structure of the database. The domination relationships between objects identified in the initial state might no longer hold in the new state. Nonetheless, computing the skylines over the entire new state/structure of the database is inefficient, as not all pairwise comparisons between the objects are necessary to be performed. In tackling the above issue, this paper proposes a solution, named Δ*Skyline*, which aims at avoiding unnecessary skyline computations when a database changes its state and structure due to a data definition operation(s) (add or remove a dimension(s)). This is achieved by identifying and retaining the prominent dominance relationships when pairwise comparisons are performed; which are then utilised in the process of computing a new skyline set. Δ*Skyline* consists of two optimisation components, namely: Δ^{+}*Skyline* which derives a new skyline set when a new dimension(s) is added to a database and Δ^{-}*Skyline* which derives a new skyline set when an existing dimension(s) is removed from a database. To make our solution more useful, it is applied on a database with incomplete data. Extensive experiments have been conducted to evaluate the performance and prove the efficiency of our proposed solution.

INDEX TERMS Multi-criteria decision making, skyline queries, incomplete database, dynamic database, pairwise comparisons.

I. INTRODUCTION

The skyline operation was introduced as an extension to the database systems by [7], that relies on the notion of Pareto dominance to filter out a set of interesting objects based on a set of evaluation criteria from a potentially large multi-dimensional set of objects. It is used in a query called as skyline query to ensure that only those objects that are not worse than any others in all the evaluation criteria which reflect the user's preferences are being selected. This leaves only those objects that are the best, most preferred objects to an arbitrary user, also known as the skyline set or pareto optimal set. Since then, it has become a vital issue in database research as it is well suited to many applications particularly those that are related to multi-preference analysis and decision making. With the rapid growth of decision support systems and the increasing size of multi-dimensional data have witnessed an abundance of skyline algorithms being 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 [23], [27], [29], [39], [40], [41], [51], incomplete data [3], [6], [13], [14], [18], [19], [20], [24], [28], [33], [49], [50], encrypted data [8], [31], and streaming data [1], [15]; while others are based on the platform being considered like distributed database [2], cloud computing [22], [32], road networks [16], and others.

Apparently, the states and/or structures of a database (multi-dimensional set of objects) change throughout its lifetime which is necessary to reflect the changing requirements,

The associate editor coordinating the review of this manuscript and approving it for publication was Genoveffa Tortora.

new functionalities, compliance to new regulations, integration with other systems, and new security or privacy measures. These changes are achieved either through a data manipulation operation(s) (insert, delete, update) or a data definition operation(s) (alter table, etc). Consequently, the changes made towards the database affect the skyline set derived earlier since the derivation of these skylines is based on the initial state of the database which may no longer valid in the new state of the database. The changes with regard to the structure of a database in particular when a new dimension(s) (criteria) is added or an existing dimension(s) is removed to/from the database, may result in some of the previously identified domination relationships between objects become invalid. This is due to the fact that objects that formerly dominate other objects may now no longer dominate the objects on the new structure of the database. This mean, whenever a database changes its structure, a new skyline set needs to be derived. Utilising the existing skyline algorithms [2], [3], [19], [20], [23], [24] would mean performing the domination analysis on the entire new structure of the database which is undoubtedly inefficient as not all pairwise comparisons between the objects need to be re-analysed. It becomes cumbersome when the number of objects involved is huge while the number of dimensions to be considered in the skyline computation is large. Although there are a few works like [13], [14], [18] that have made attempts to avoid unnecessary computation of skylines when changes are made towards a database, their emphasis is on changes that are due to new object(s) is inserted into the database or an existing object(s) is deleted from the database with a fixed and predetermined set of dimensions (criteria).

This paper proposes a solution that attempts to avoid unnecessary computation of skylines, i.e. unnecessary pairwise comparisons between objects, when a new skyline set needs to be identified due to changes made towards a database. Our solution, named Δ*Skyline*, consists of two main phases, namely: *Phase* I – capturing the dominance relationships based on the initial database and *Phase* II – computing the new skyline set when a database changes its state and structure due to a data definition operation(s). Fig. 1 presents the Δ*Skyline* framework. In [13], we proposed a solution that derives a new skyline set when a database changed its state due to a data manipulation operation(s); in which a new object(s) is inserted into the database or an existing object(s) is deleted or updated from the database. In this paper, we dealt with operations that either add a new dimension(s) to a database or remove an existing dimension(s) from a database which will not only change the state of the database but also its structure. In order to achieve our main aim, i.e. avoiding unnecessary computations of skylines, the dominance relationships between objects that are identified when pairwise comparisons are performed are retained to be utilised during the process of identifying a new skyline set. However, keeping track of each dominance relationship is unwise as not only it will incur unnecessary storage cost, also not all the dominance relationships will be

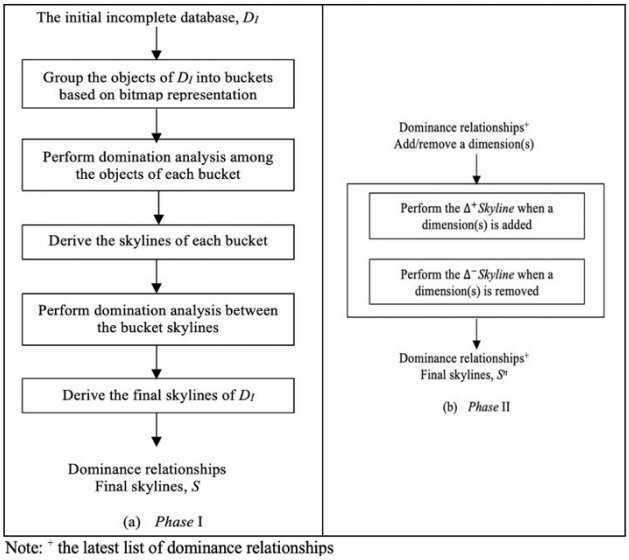

FIGURE 1. The Δ*Skyline* framework.

utilised in the subsequent processes of skyline computations. Hence, besides the main issue of optimisation, identifying the prominent dominance relationships among all possible dominance relationships is another issue to be dealt with. To make our solution more useful, we consider a database with incomplete data. This is in line with today's era, where most real-world applications often deal with data that are partly missing or incomplete. The existence of incomplete data in a database is due to many reasons such as negligence in data entry, inaccurate data from heterogeneous data sources, and integrating heterogeneous schemas [30], [36], [43], [45]. There are several works that have made attempts to optimise the skyline computation over an incomplete database such as [2], [3], [6], [19], [20], [23], [24], [28], [43], [49], and [50], however these works do not consider the possibilities that the database might change its structure, hence strictly relying on the assumption that the database is always static. Thus, the skyline set derived by these works is based on a fixed set of dimensions (criteria). Although their solutions can be applied directly on the new structure of the database to derive a new skyline set, this will incur unnecessary computation of skylines since their solutions do not incorporate mechanism that is able to analyse the parts of the database that are affected by the changes.

In general, the main contributions of this work are briefly described as follows:

- We have formally introduced the problem of computing skylines when a database changes its state and structure and justify the significance of addressing the problem.
- We have proposed an efficient solution, named Δ*Skyline* that attempts to avoid unnecessary skyline computations when changes made towards an incomplete database are due to a data definition operation(s), i.e. adding a new dimension(s) to a database or removing an existing

dimension(s) from a database which will not only change the state of the database but also its structure.
- We have devised a mechanism to identify and retain the prominent dominance relationships to be utilised during the computation of new skyline set that assists in excluding the unnecessary pairwise comparisons between objects when changes are made towards a database. In this regard, three main lists have been designed, namely: *Domination Analysis List* (*DAL*), *Dominating Object List* ($\succ OL$), and *Dominated Object List* ($\not\succ OL$), that keep track of the results of domination analysis, objects that dominate other objects, and objects that are dominated by other objects, respectively.
- We have conducted extensive experiments to prove Δ*Skyline*'s capability in deriving a skyline set after changes are made towards a database, which are reflected in the state as well as the structure of the database.

The rest of the paper is structured as follows. In Section II, the previous works that are related to computing skylines over complete as well as incomplete databases are presented. In Section III, the necessary definitions and notations, which are used throughout the paper, are set out. Section IV elaborates our proposed approach in handling the computation of skyline set given an incomplete database that changes its state and structure. A running database example is also given to clarify the phases of the proposed approach. The experimental results are demonstrated in Section V. Conclusion and further research direction are depicted in the final section, Section VI.

II. RELATED WORKS

Since the introduction of skyline operator by [7], a significant number of skyline algorithms have been proposed, mostly attempt to resolve the optimisation problem that is mainly associated with the reduction in the processing time as well as the number of pairwise comparisons that needs to be performed during the skyline computations. These algorithms can be categorised into two distinct groups based on the approach used in processing the skyline query, namely: index-based algorithms and non-index based algorithms. Unlike the non-index based algorithms, the index-based algorithms requires pre-computed indexes on data to avoid accessing the entire data. Although the index-based algorithms have better performance, they suffer from curse of dimensionality. Hence, many variants of skyline algorithms have evolved; among the notable algorithms include *Divide-and-Conquer* (*D&C*), *Block Nested Loop* (*BNL*) [7], *Bitmap* and *Index* [44], *Nearest Neighbor* (*NN*) [26], *Branch and Bound Skyline* (*BBS*) [37], [38], *Sort Filter Skyline* (*SFS*) [11], *Linear Elimination Sort for Skyline* (*LESS*) [17], and *Sort and Limit Skyline algorithm* (*SaLSa*) [5]. Principally, the *BNL* algorithm works by repeatedly reading a set of objects and eliminating those objects that are dominated by other objects in the data set. Meanwhile, the *D&C* algorithm proposed by [7], divides the data set into partitions to allow domination analysis to be performed on individual partition in which local skylines are derived. These local skylines are then compared to each other to derive the global skylines of the data set. *Bitmap* [44] is an example of an index-based algorithm which encodes all the required information based on a bitmap structure prior to the skyline computation. Based on the bitmap structure, whether or not an object belongs to the skylines is determined without the necessity to scan the entire data set. On the other hand, *Index* [44] algorithm, also an index-based algorithm, partitions the data set into d ordered list where d is the set of dimensions considered in the skyline computation while B-tree is used to index the objects. In each list, objects are organised in batches and sorted in certain order based on the dimension. Local skylines are computed for each batch while global skylines are computed among these local skylines. Nonetheless, the *NN* algorithm proposed by [26], utilises the R^*-tree index structure to eliminate objects with emphasis on avoiding redundant dominance checks. With similar aim as *NN*, the *BBS* [37], [38], algorithm which applies the nearest neighbor search techniques requires traversing the R^*-tree only once; hence improves the *NN* algorithm which traverses the R^*-tree several times. The *BNL* algorithm is later improved by [11]; the *SFS* algorithm utilises a monotone preference function f to presort the objects of a data set in an ascending order. An object that is visited earlier is said to be dominating the objects that appeared later in the list. Later, an optimised version of *SFS* is proposed by [17] named *LESS* with an attempt to position the killer-dominant objects at the beginning of the sorted data set. This is achieved by applying the entropy scoring function. In an attempt to further improve the performance of *SFS* and *LESS*, [5] proposed an ideal algorithm named *SaLSa* that limits the number of objects to be read and compared by utilising a threshold value while exploiting the values of a monotone scoring (limiting) function. Although these algorithms solve the optimisation problem, they are designed with a rigid assumption that the data set is complete and certain.

Recently, attention has been given to resolving issues related to the uncertainty of data in a database. Data uncertainty is defined as the degree to which data are inaccurate, imprecise, untrusted, unknown or incomplete. The works by [23], [27], [29], [39], [40], [41], and [51] for instance strived to solve issues of skyline analysis on uncertain data. Pei *et al.* [39] has introduced the notion of probabilistic skyline in the context of uncertain data in discrete domains where each object is associated with probability distributions over a set of possible values called instances. This pioneering work has then inspired several other works in the same context like [29], [40], and [51]. Meanwhile, the works by [23], [27], and [41], dealt with uncertainty in continuous domains where uncertain data are represented as continuous range of values, in which the precise values are not known during the skyline computation. Besides, several attempts have been made to tackle the issues related to the incompleteness of data in a database. The works by [28], [49], and [50], for instance focus on techniques to rank the skyline results in which some of

the results are incomplete. While, [4] and [19] attempt to derive a better set of skyline results by integrating both top-k and skylines. Others like [33] study the problem of skyline queries over incomplete data with crowdsourcing, while the work by [34] focuses on the problem of k-dominant skyline queries on incomplete data. To the best of our knowledge only the works by [3], [6], and [24] have made attempts to tackle the issues of skyline computation over an incomplete database which are further elaborated below.

The early research work that deals with the issues related to skyline computation over incomplete data is conducted by [24], in which two algorithms are proposed, namely: *Bucket* and *ISkyline*. The *Bucket* algorithm relies on the bitmap representation of each object to group the objects into buckets. Local skylines are then identified by utilising the conventional skyline algorithm. Finally, the local skylines of each bucket are compared to each other to derive the final skylines. Meanwhile, the *ISkyline* algorithm works in a similar fashion as *Bucket* algorithm, with two additional optimisation techniques to reduce the number of local skylines. In *SIDS* [6], the input data set is pre-sorted in a non-increasing order for each dimension, to determine the processing order of the objects. The proposed approach chooses one of the dimensions in a round-robin fashion and the object with the best value in that dimension is chosen for processing. The algorithm initially considers all objects in the data set as candidate skylines and then iteratively removes dominated objects from the candidate set. If an object has not been pruned yet and has been processed k times, where k is the count of complete dimensions for the object, then it is determined to be a skyline and can be returned immediately. The reason is any object with k complete dimensions can be dominated in at most k dimensions. Similar to *ISkyline* algorithm, *Incoskyline* [3], works by clustering the objects into related buckets, in which local skylines are then identified. *Incoskyline* which is based on a heuristic approach, derives a set of virtual skylines named k-dom from the local skylines to eliminate the local skylines of a cluster.

Despite the fact that these works have extensively solved the issues related to incompleteness of data, however they are designed with a rigid assumption that the dimensions of the database or criteria to be considered during the skyline computation are predetermined and fixed in which any changes towards the dimension (criteria) to be considered would require re-examining the entire database based on the new set of dimensions (criteria). There are algorithms which are proposed specifically for such situation but most of them are based on top-k. For instance, the work by [25] focuses on top-k and top-k dominating and reviews algorithms for evaluating continuous preference queries under the sliding window streaming model. Also, a k-dominant skyline algorithm has been presented in [12]. In their work when the data set is changed, the existing k-dominant skylines are compared to the new k-dominant objects to derive the results [12].

III. DEFINITIONS AND NOTATIONS

In this section, we present the necessary definitions and introduce the notations that are used throughout this paper. First, we explain the concept of skyline through an example. Then, we give the general definitions that have been defined either formally or informally in the literature [2], [6], [7], [13], [19], [24] based on the notations used in this paper (i.e. *Definition* 1 till *Definition* 6). This is then followed with specific definitions that are related to our work. Examples are provided where necessary to further clarify the definitions. Finally, a formal definition of the problem addressed in this paper is put forward.

A. MOTIVATION EXAMPLE

Consider a user who is planning to buy an apartment with the following main features: apartment that is cheapest and nearest to the city centre. Fig. 2(a) presents a toy example for this scenario. In this example, there are 6 objects representing 6 distinct apartments labelled as a, b, c, d, e, and f with each object having two attributes, namely: *price* and *distance* that are used as the evaluation criteria. More often than not, apartments that are in prime areas close to the city centre are more expensive as compared to those which are far away from the city centre. In other words, *price* and *distance* are two conflicting criteria which implies that the chances to find an apartment that meets both preferences are low. Utilising the skyline operation proposed by [7] on the given example, we have the following: object e dominates object b, while object c dominates objects d and f. Meanwhile object a neither dominates nor being dominated by any other objects. The results of the domination analysis are as shown in Fig. 2(a). Eventually, based on our toy example, the user is left with the following filtered apartments: a (apartment with the minimum distance to the city centre), c (apartment with the minimum price), and e (apartment with price cheaper than a and distance nearer than c). Undoubtedly, to derive the list of filtered apartments, objects are compared to each other based on the evaluation criteria being considered in a pairwise manner. The skyline computation becomes expensive as the number of evaluation criteria as well as the number of objects to be considered are huge. Hence, one of the properties assumed by most skyline algorithms is the transitivity in the dominance relation in which various ways of data pruning and indexing can be exploited to reduce the number of pairwise comparisons [24]. For instance, if it is known that object c dominates object f while object f dominates object d, hence without further comparison we can conclude that object c dominates object d.

Proceeding with our toy example, assume that a new dimension, *rating* which represents the service rating of the apartments, is to be considered as shown in Fig. 2(b). Here, we assume that the higher the value of rating the better is the service. Utilising the skyline operation proposed by [7] on the new state presented in Fig. 2(b), we have the following: object e still dominates object b, while object c

Object	Price	Distance
a	200	0.5
b	150	2
c	25	3.5
d	125	4.5
e	100	1.5
f	75	4

Domination analysis

Dominating	Dominated
e	*b*
c	*d*, *f*

(a) A set of objects with two dimensions and the results of domination analysis

Object	Price	Distance	Rating
a	200	0.5	5
b	150	2	4
c	25	3.5	2
d	125	4.5	2
e	100	1.5	4
f	75	4	5

Domination analysis

Dominating	Dominated
e	*b*
c	*d*

(b) A set of objects with an additional dimension and the results of domination analysis

Object	Price	Distance	Square Feet
a	200	0.5	500
b	150	2	600
c	25	–	600
d	–	4.5	800
e	100	1.5	700
f	75	4	–

Domination analysis

Dominating	Dominated
e	*b*
c	*f*
f	*d*
d	*c*

(c) A set of objects with missing values and the results of domination analysis

FIGURE 2. Example of skylines.

dominates object d and no longer dominates object f. Meanwhile object a neither dominates nor being dominated by any other objects. The results of the domination analysis are as shown in Fig. 2(b). The skylines of this example are objects a, c, e, and f. Here, the following can be observed: (i) object that dominates other objects based on the previous evaluation criteria (*price* and *distance*) is still part of the skyline set, e.g. c and e; (ii) object that is dominated might have chances to be a skyline if its value based on the new criterion (*rating*) is better than the object that dominates it, e.g. f; and (iii) object that neither dominates nor being dominated by any other objects is still the skyline, e.g. a. The same can be visualised if a dimension(s) is to be removed from a database (working from Fig. 2(b) to Fig. 2(a)). An important remark is that the computation of the new skyline set should not repeat the domination analysis that has been conducted earlier.

Meanwhile, objects may have missing values in one or more of their dimensions. However, the transitive dominance relation as discussed earlier might no longer hold. This is demonstrated in Fig. 2(c). Here, object c dominates object f, object f dominates object d, while object d dominates object c. As a result, these objects are filtered out and none of them are considered as skylines.

B. FUNDAMENTAL PROPERTIES OF SKYLINES

This section presents the general definitions that are related to skyline query processing. For *Definition*1 till *Definition* 6, we assume that the following is given: a database $D = \{o_1, o_2, \ldots, o_w$ where o_i is the ith object with each object associated with m dimensions (m criteria) denoted by $d = \{d_1, d_2, \ldots, d_m\}$ that are to be considered in the skyline computation.

When a pair of objects is compared to each other based on the values of their dimensions, it can be concluded that either one is being dominated by the other or both do not dominate each other. While an object that is not being dominated by any other objects in the database is said to be the skyline of the database. These are formally defined in Definition 1 and Definition 2, respectively.

Definition 1 *Dominance Relationship:* An object $o_i \in D$ is said to dominate an object $o_j \in D$ where $i \neq j$ denoted by $o_i \succ o_j$ if and only if the following condition holds: $\forall d_k \in d, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d, o_i.d_l > o_j.d_l$. For instance, object $e(100, 1.5)$ is said to dominate object $b(150, 2)$ denoted by $e \succ b$ as object e is better than object b in both dimensions, meanwhile object $b(150, 2)$ and object $c(25, 3.5)$ are said to not dominate each other as object b is better than object c only in the second dimension and worse in the first dimension and similarly object c is better than object b only in the first dimension and worse in the second dimension. This is denoted by $b \not\succ c$ and $c \not\succ b$.

Definition 2 *Skylines:* An object $o_i \in D$ is a skyline of D if there are no other objects $o_j \in D$ where $i \neq j$ that dominates o_i. In this paper, the symbol S is used to denote the skyline set of D. For instance, object $c(25, 3.5)$ is one of the skylines in S as it is not dominated by other objects in the set.

Since this paper assumed that a database may contain objects with missing values, it is important to define the scope of missing values as used in this paper. Also, it is unrealistic to determine the dominance relationship between objects not having values on common attributes. These are further clarified by Definition 3 and Definition 4, respectively.

Definition 3 Incomplete Database: A database D is said to be incomplete denoted by D_I if and only if $\exists o_i \in D, o_i \cdot d_k = {}' - {}'$ where $d_k \in d$ and $' - '$ denotes a missing value (null value); otherwise the database is said to be complete. For instance, the data set given in Fig. 2(c) is said to be incomplete as the objects c, d, and f have missing values in one of the dimensions. Even though $o_i(-, -, -)$ satisfies the Definition 3, in our work we do not deal with the incompleteness of data that can be viewed as objects that are missing as a whole.

Definition 4 Comparable: The objects $o_i \in D$ and $o_j \in D$ where $i \neq j$ are said to be comparable if and only if they are either complete or they have missing values in the same

dimension(s). Here, each object is associated with a bitmap representation in which the bit 1 is used to represent non missing value, i.e $o_i \cdot d_k \neq {}' - {}'$; while the bit 0 otherwise, i.e. $o_i \cdot d_k = {}' - {}'$. With this notation those objects with the same bitmap representation are said to be comparable. For instance the objects a, b, and e in Fig. 2(c) are said to be comparable as they are complete and their bitmap representation is 111. Meanwhile objects c and d are said to be incomparable as their bitmap representations are 101 and 011, respectively.

A simple mechanism to identify whether a pair of objects is comparable or not is by analysing their bitmap representations as presented in Definition 5; meanwhile the dominance relationships between these objects can be determined as explained in Definition 6.

Definition 5 Revised Bitwise: The objects $o_i \in D$ and $o_j \in D$ where $i \neq j$ with different bitmap representations are said to be comparable on their *revised bitwise* if it is not equal to 0, which is obtained by performing the AND operation on the bitmap representations of o_i and o_j. For instance, the objects a and c in Fig. 2(c) with bitmap representations 111 and 101, respectively are said to be comparable on their *revised bitwise* as the result of performing the AND operation on 111 and 101 is 101 which is not equal to 0.

Definition 6 Dominance Relationship on the Revised Bitwise: An object $o_i \in D$ is said to dominate an object $o_j \in D$ where $i \neq j$ on the *revised bitwise* denoted by $o_i \succ o_j$ if and only if the following condition holds: $\forall d_k \in d', o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d', o_i.d_l > o_j.d_l$ where $d' \subset d$ and d' is a set of dimensions whose revised bitwise representation is 1. For instance, object c in Fig. 2(c) with bitwise representation 101 is said to dominate object f with bitwise representation 110, i.e. $c \succ f$, as c is better than f in the first dimension.

C. EXTENDED PROPERTIES OF SKYLINES

This section presents the definitions that are specific to our work. For *Definition* 7 until *Definition* 12, we assume the following: A database,[1] D, with m dimensions, $d^m = \{d_1, d_2, \ldots, d_m\}$ to be considered in the skyline computation denoted as D^m with w objects, $D^m = \{o_1, o_2, \ldots, o_w\}$. Also, we assume that the operations to add a dimension(s) or remove a dimension(s) to/from the database are well-formed.

In this work, the structure of a database is defined mainly based on the number of dimensions it contains. The structure of a database is not fixed throughout its lifetime as a new dimension(s) might need to be added to the database or an existing dimension(s) might have to be removed from the database. Definitions 7, 8, and 9 defined the structure of a database as used in this work.

Definition 7 Structure of a Database: The database D^m is said to be in a new structure with n dimensions, $d^n = \{d_1, d_2, \ldots, d_n\}$ denoted as D^n where $n > m$ or $n < m$ due to either (i) a new dimension(s) is added to D^m or (ii) an existing dimension(s) is removed from D^m. For instance, the initial set of objects in Fig. 2(a) denoted as D^2 with $d^2 = \{price, distance\}$ is in a new structure presented in Fig. 2(b) denoted as D^3 with $d^3 = \{price, distance, rating\}$.

Definition 8 Adding a New Dimension(s): Given a set of new dimensions to be added, $d_{<add>} = \{d_a, d_{a+1}, \ldots, d_{a+b}\}$ to D^m, the new structure of the database is denoted as D^n with $n = |d| + |d_{<add>}|$. We use the notation D^{n-m} to denote the part of the database with $d_{<add>}$ dimensions. Also, note that $d_{<add>}$ contains $|d_{<add>}|$ new criteria to be considered in the skyline computation. For instance, the $d_{<add>} = \{rating\}$ is added to D^2 (Fig. 2(a)) to yield D^3 as presented in Fig. 2(b).

Definition 9 Removing an Existing Dimension(s): Given a set of dimensions to be removed, $d_{<remove>} = \{d_r, d_{r+1}, \ldots, d_{r+s}\}$ from D^m, the new structure of the database is denoted as D^n with $n = |d| - |d_{<remove>}|$. We use the notation D^{m-n} to denote the part of the database with $d_{<remove>}$ dimensions. Also, note that $d_{<remove>}$ is the set of existing criteria that is no longer relevant in the skyline computation. For instance, the $d_{<remove>} = \{square\ feet\}$ is removed from D^3 (Fig. 2(c)) to yield D^2 as presented in Fig. 2(a).

The following definitions extend the definition of dominance relationship given in Definition 1 based on the new structure of a database as defined in Definition 7.

Definition 10 Dominance Relationship of D^n Based on $d_{<add>}$: An object $o_i \in D^n$ is said to dominate an object $o_j \in D^n$ where $i \neq j$ denoted by $o_i \succ o_j$ if and only if the following condition holds: $\forall d_k \in d^n, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d^n, o_i.d_l > o_j.d_l$ or given $d^m = \{d_1, d_2, \ldots, d_m\}$ and $d_{<add>} = \{d_a, d_{a+1}, \ldots, d_{a+b}\}$,

1) $\forall d_k \in d^m, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d^m, o_i.d_l > o_j.d_l$ and $\forall d_k \in d_{<add>}, o_i.d_k \geq o_j.d_k$ or
2) $\forall d_k \in d^m, o_i.d_k \geq o_j.d_k$ and $\forall d_k \in d_{<add>}, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d_{<add>}, o_i.d_l > o_j.d_l$.

For instance, assume that D^2 is the set of objects shown in Fig. 2(a) and D^3 is the set of objects shown in Fig. 2(b) after adding $\{rating\}$ to D^2. Object $c(25, 3.5)$ is said to dominate object $f(75, 4)$ over D^2 denoted by $c \succ f$ as object c is better than object f in both dimensions. However, object $c(25, 3.5, 2)$ and object $f(75, 4, 5)$ are said to not dominate each other over D^3 as although object c is better than object f with regard to *price* and *distance* but it is worse than f based on the new criterion, *rating*.

Definition 11 Dominance Relationship of D^n based on $d_{<remove>}$: An object $o_i \in D^n$ is said to dominate an object $o_j \in D^n$ where $i \neq j$ denoted by $o_i \succ o_j$ if and only if the following condition holds: $\forall d_k \in d^n, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d^n, o_i.d_l > o_j.d_l$.

The following definition is an extension to the skyline definition given in Definition 2. Here, the skylines are computed based on the new structure of a database.

Definition 12 Skylines of D^n: An object $o_i \in D^n$ is a skyline of D^n if there is no other objects $o_j \in D^n$ where $i \neq j$ that dominates o_i. We use the notation S^n to denote the skyline set of D^n. For instance, the skyline set of D^2 given in Fig. 2(a),

[1]Without loss of generality, the term database covers both complete and incomplete database.

TABLE 1. Number of pairwise comparisons based on the type of data definition operation.

Case/Operation	Best-case		Average-case		Worst-case	
	S^m	S^n	S^m	S^n	S^m	S^n
Add	$m(w-1)$	$n(w-1)$ where $n > m$	$\frac{m(w-1)\left(\frac{w}{2}+1\right)}{2}$	$\frac{n(w-1)\left(\frac{w}{2}+1\right)}{2}$ where $n > m$	$m\left\lceil\frac{w(w-1)}{2}\right\rceil$	$n\left\lceil\frac{w(w-1)}{2}\right\rceil$ where $n > m$
Remove	$m(w-1)$	$n(w-1)$ where $n < m$	$\frac{m(w-1)\left(\frac{w}{2}+1\right)}{2}$	$\frac{n(w-1)\left(\frac{w}{2}+1\right)}{2}$ where $n < m$	$m\left\lceil\frac{w(w-1)}{2}\right\rceil$	$n\left\lceil\frac{w(w-1)}{2}\right\rceil$ where $n < m$

$S^2 = \{a, c, e\}$ while the skyline set of D^3 given in Fig. 2(b), $S^3 = \{a, c, e, f\}$.

For simplicity purposes, we use the notations o_{i^m}, o_{i^n}, $o_{i^{n-m}}$, and $o_{i^{m-n}}$ to denote the object o_i with dimensions based on the structure D^m, D^n, D^{n-m}, and D^{m-n}, respectively.

D. PROBLEM FORMULATION

In the following we show the significant of avoiding unnecessary skyline computations by analysing the number of pairwise comparisons for each of the operation considered in this paper. Here, we assume a worst-case scenario while both the best-case and average-case scenarios are presented in Table 1. The number of pairwise comparisons performed to derive the skyline set, S^m, over D^m is given by the following equation:

$$\alpha = m\left\lceil\frac{w(w-1)}{2}\right\rceil \tag{3.1}$$

where m is the number of dimensions and w is the number of objects. For instance, $\alpha = 30$ based on D^2 shown in Fig. 2(a).

When $d_{<add>} = \{d_a, d_{a+1}, \ldots, d_{a+b}\}$ is added to D^m, based on Equation 3.1 the number of pairwise comparisons performed to derive the skyline set, S^n, over the new structure D^n is $n\left\lceil\frac{w(w-1)}{2}\right\rceil$ where n is the number of dimensions with $n = |d^m| + |d_{<add>}|$, i.e. $n > m$, and w is the number of objects. However, S^m is derived by comparing the w objects based on d^m, while S^n is derived by comparing the w objects based on d^m and $d_{<add>}$. Hence, the number of unnecessary pairwise comparisons is $(n-m)\left\lceil\frac{w(w-1)}{2}\right\rceil$. For instance, $\alpha = 45$ based on D^3 shown in Fig. 2(b) in which 15 pairwise comparisons are unnecessary.

Hence, given a sequence of k add operations, $\{d_{<add>_1}, d_{<add>_2}, \ldots, d_{<add>_k}\}$, where $d_{<add>_i}\}$ is a set of new dimensions to be added to D^m in the ith sequence, i.e. $d_{<add>_i} = \{d_{a_i}, d_{a+1_i}, \ldots, d_{a+b_i}\}$, the number of pairwise comparisons performed to derive the skyline set, S^n, over the new structure D^n is $\sum_{i=1}^{k} n_i\left\lceil\frac{w(w-1)}{2}\right\rceil$ where n_i is the number of dimensions to be added in the ith sequence with $n > m$.

When $d_{<remove>} = \{d_r, d_{r+1}, \ldots, d_{r+s}\}$ is removed from D^m, the number of pairwise comparisons performed to derive the skyline set, S^n, over the new structure D^n is $n\left\lceil\frac{w(w-1)}{2}\right\rceil$ where n is the number of dimensions with $n = |d^m| - |d_{<remove>}|$, i.e. $m > n$, and w is the number of objects. However, S^m is derived by comparing the w objects based on d^m, while S^n is derived by comparing the w objects based on $d^m - d_{<remove>}$. Hence, the number of unnecessary pairwise comparisons is $n\left\lceil\frac{w(w-1)}{2}\right\rceil$. For instance, $\alpha = 45$ based on D^3 shown in Fig. 2(b). Assume that *rating* is removed from D^3 hence the number of unnecessary pairwise comparisons is 30.

Hence, given a sequence of k remove operations, $\{d_{<remove>_1}, d_{<remove>_2}, \ldots, d_{<remove>_k}\}$, where $d_{<remove>_i}$ is a set of existing dimensions to be removed from D^m in the ith sequence, i.e. $d_{<remove>_i} = \{d_{r_i}, d_{r+1_i}, \ldots, d_{r+s_i}\}$, the number of pairwise comparisons performed to derive the skyline set, S^n, over the new structure D^n is $\sum_{i=1}^{k} n_i\left\lceil\frac{w(w-1)}{2}\right\rceil$ where n_i is the number of dimensions to be removed in the ith sequence with $m > n$.

For mixed operations that involve both operations, add and remove, the number of pairwise comparisons performed to derive the skyline set, S^n, over the new structure D^n is simply obtained by the following equation: $\sum_{i=1}^{k} n_i\left\lceil\frac{w(w-1)}{2}\right\rceil$ where n_i is the number of dimensions to be added or removed in the ith sequence.

The problem addressed by this paper is formulated as follows:

Problem Formulation: Given an initial incomplete database, D^m, with m dimensions $d^m = \{d_1, d_2, \ldots, d_m\}$ and the skyline set, S^m, i.e. $S^m \subseteq D^m$. It is known that S^m is a set of objects that is not dominated by other objects in $D^m - S^m$, say $\neg S^m$. When there are changes in the structure/state of D^m due to either (i) a new dimension(s), $d_{<add>}$, is added to D^m or (ii) an existing dimension(s), $d_{<remove>}$, is removed from D^m, how can we efficiently compute the new skyline set, S^n, of the new structure of the database denoted as D^n without the necessity to analyse the entire database, D^n, to avoid unneccesary computational cost?

Table 2 summarises the symbols and notations used throughout this paper.

IV. THE PROPOSED FRAMEWORK

This section presents our proposed solution, named Δ*Skyline*, which is designed with the main aim at avoiding unnecessary skyline computations when changes are made towards a database owing to a data definition operation(s). As a result of these changes, the skyline set derived earlier is no longer valid. Nonetheless, it is unwise to identify a new skyline

TABLE 2. List of symbols/notations.

Symbols/Notations	Remarks
$D = \{o_1, o_2, \ldots, o_w\}$; $D^m = \{o_1, o_2, \ldots, o_w\}$; $D^n = \{o_1, o_2, \ldots, o_w\}$	A database with a set of w objects with implicit m dimensions; an initial database with m dimensions and a set of w objects; a database with n dimensions and a set of w objects that represents the new structure of a database, respectively
D_I	Incomplete database
$d = \{d_1, d_2, \ldots, d_m\}$; $d^m = \{d_1, d_2, \ldots, d_m\}$	A set of m dimensions
$d_{<add>} = \{d_a, d_{a+1}, \ldots, d_{a+b}\}$	A set of new dimensions to be added
$d_{<remove>} = \{d_r, d_{r+1}, \ldots, d_{r+s}\}$	A set of existing dimensions to be removed
$\{d_{<add>_1}, d_{<add>_2}, \ldots, d_{<add>_k}\}$	A sequence of k add operations
$d_{<add>_i} = \{d_{a_i}, d_{a+1_i}, \ldots, d_{a+b_i}\}$	A set of new dimensions to be added in the ith sequence
$\{d_{<remove>_1}, d_{<remove>_2}, \ldots, d_{<remove>_k}\}$	A sequence of k remove operations
$d_{<remove>_i} = \{d_{r_i}, d_{r+1_i}, \ldots, d_{r+s_i}\}$	A set of existing dimensions to be removed in the ith sequence
$o_i \succ o_j$	Object o_i dominates object o_j
B_i	The ith bucket
$S; S^m;\ S^n; S_{B_i}; Sc$	Skyline set; skyline set of D^m; skyline set of D^n; bucket skylines; candidate skylines, respectively
α	The number of pairwise comparisons

set by analysing the entire database after the changes are made, since not all pairwise comparisons between objects need to be re-analysed. Intuitively, this can be achieved by keeping track the dominance relationships between objects that are identified during the computation of skylines before the database is changed. When changes are made, these dominance relationships are re-examined and only those objects that are affected are further analysed. Consequently, this leads to a significant reduction in the number of pairwise comparisons and processing time in the subsequent computation of skylines.

Δ*Skyline* consists of two phases, that are: *Phase* I – the main aim of this phase is to keep track of the dominance relationships between objects while the skyline set is derived based on the initial incomplete database; and *Phase* II – a new skyline set is derived due to the changes that are made towards the database. To avoid unnecessary skyline computations during *Phase* II, the dominance relationships captured in *Phase* I are utilised and updated accordingly to incorporate any new dominance relationships realised in *Phase* II. These two phases are elaborated in the following subsections. In order to better explain the steps that are involved in each phase, the sample of database shown in Fig. 3 will be used throughout this section. The sample database D_I contains 15 objects, $D_I = \{o_1, o_2, \ldots, o_{15}\}$ with 3 dimensions, $d = \{d_1, d_2, d_3\}$, that are used as the evaluation criteria in deriving the skyline set. For simplicity, we limit the number of missing values of each object to one. Nevertheless, Δ*Skyline* can handle various number of missing values which is exhibited in the experiments that we have conducted.

Object	d_1	d_2	d_3
o_1	–	5	3
o_2	–	1	3
o_3	7	–	6
o_4	–	3	2
o_5	2	3	–
o_6	6	4	–
o_7	5	4	–
o_8	3	–	5

Object	d_1	d_2	d_3
o_9	6	–	7
o_{10}	–	6	6
o_{11}	3	3	–
o_{12}	–	3	6
o_{13}	3	6	–
o_{14}	2	1	–
o_{15}	5	–	7

FIGURE 3. Example of an incomplete database, D_I.

A. PHASE I

Given an initial incomplete, database, D_I, the *Phase*I derives a skyline set, S, and maintains the results of domination analysis to be utilised by *Phase* II. It is performed only once since the main aim of this phase is to keep track and maintain the dominance relationships between objects which are then used to identify the pairwise comparisons that ought to be performed between the affected objects when changes are made towards the database, D_I. We have devised five steps to be conducted in this phase, namely: (i) group the objects of D_I into buckets based on the bitmap representation, (ii) perform domination analysis among the objects of each bucket, (iii) derive the skyline of each bucket, (iv) perform domination analysis between the bucket skylines, and (v) derive the final skylines of D_I. Each step is elaborated further in the following paragraphs.

Step 1 *Group the Objects of* D_I *Into Buckets Based on the Bitmap Representation:* This step is performed only if the initial database contains objects with missing values. Here, we are assuming an initial incomplete database, D_I. As explained earlier, objects with missing values may cause cyclic dominance and complication in preserving the transitivity property of skylines which results in none of the objects involved in the cycle domination can be considered as skylines. This can be represented as follows: $o_i \succ o_j, o_j \succ o_k$, and $o_k \succ o_i$. Hence to resolve this issue, each object, $o_i \in D_I$, is associated with a bitmap representation (see *Definition* 4 *Comparable*). The objects with the same bitmap representation are grouped into the same bucket. By doing this, each bucket contains a collection of objects with missing values in the same dimension(s). The number of buckets derived depends on the number of distinct bitmap representations that can be formed given the database D_I. For instance, based

on the sample database given in Fig. 3, the distinct bitmap representations formed are 011, 101, and 110. This technique, also known as bucketing/clustering/grouping is a well-known technique that has been utilised by other previous works like [3] and [24]. It simply works by comparing the bitmap representation of an object, say o_i, with the bitmap representation of existing buckets, say B_j. If they have the same bitmap representation, then the object o_i is said to belong to the bucket B_j. Otherwise, a new bucket, say B_k, is created based on the bitmap representation of object o_i. Fig. 4 shows the results of performing the bucketing technique over the sample database given in Fig. 3. Here, the set of buckets formed, $B = \{B_1, B_2, B_3\}$, with each bucket with a distinct bitmap representation.

Object	d_1	d_2	d_3
o_1	–	5	3
o_2	–	1	3
o_4	–	3	2
o_{10}	–	6	6
o_{12}	–	3	6

B_1 (011)

Object	d_1	d_2	d_3
o_3	7	–	6
o_8	3	–	5
o_9	6	–	7
o_{15}	5	–	7

B_2 (101)

Object	d_1	d_2	d_3
o_5	2	3	–
o_6	6	4	–
o_7	5	4	–
o_{11}	3	3	–
o_{13}	3	6	–
o_{14}	2	1	–

B_3 (110)

FIGURE 4. **The results of Bucketing Technique.**

Step 2 *Perform Domination Analysis Among the Objects of Each Bucket:* Once the bucketing technique has been conducted over the objects of a given database, D_I, and buckets have been formed, $B = \{B_1, B_2, \ldots, B_l\}$, domination analysis is then performed. Here, each object within the same bucket is compared against each other in a pairwise manner. Since objects in the same bucket have the same bitmap representation, then they are said to be comparable (*See Definition* 4 *Comparable*). However, for bucket with bitmap representation containing at least one bit 0, the dominance relationship defined in *Definition* 1 is modified as follows as the set of dimensions used in the comparisons is no longer d.

Definition 13 Dominance Relationship of a Bucket: An object $o_i \in B_k$ is said to dominate an object $o_j \in B_k$ where $i \neq j$ denoted by $o_i \succ o_j$ if and only if the following condition holds: $\forall d_k \in d', o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d', o_i.d_l > o_j.d_l$ where $d' \subset d$ and d' is a set of dimensions of B_k with bitwise representation having the value 1. In this regard, d' is the set of dimensions (criteria) that is considered in determining the dominance relationships between objects of the bucket B_k while $d'' = d - d'$ is not considered as the values over the d" for all objects in B_k are missing.

For instance, the dominance relationships between objects of bucket B_1 with bitmap representation 011 are identified based on $d' = \{d_2, d_3\}$ while d" $= \{d_1\}$ is not considered; meanwhile the d' for B_2 is $\{d_1, d_3\}$ and d" $= \{d_2\}$. Hence, every bucket refers to a different set of dimensions, d', during the domination analysis. Meanwhile, objects that are dominated by other objects are removed from the bucket.

The results of the domination analysis are captured and saved into a list called *Domination Analysis List*(*DAL*). The *DAL* which keeps track of every dominance relationship that occurs between objects (See *Definition* 13 *Dominance Relationship*) has the following structure <*Object Dominating*, *Object Dominated*> where *Object Dominating* represents those objects that dominate other objects in the bucket while *Object Dominated* represents those objects that are being dominated. For instance, if $o_1 \succ o_2$, then $DAL = \{< o_1, o_2 >\}$. While if $o_{10} \succ o_1$ and $o_{10} \succ o_{12}$, then the list is expanded as follows: $DAL = \{< o_1, o_2 >, < o_{10}, \{o_1, o_{12}\}>\}$ which states that o_1 is better than o_2 and similarly o_{10} is better than o_1 and o_{12} in all the dimensions (evaluation criteria) being considered. This list becomes realistic once changes are to be made to the set of dimensions being considered in deriving the skyline set. For instance, if a new dimension (criterion) is to be added (considered), then performing the domination analysis on the entire bucket/database is no longer necessary. It is sufficient to compare the pairs of dominating and dominated objects saved in the *DAL* based on the new given dimension. Fig. 5(a) shows the entries of *DAL* based on bucket B_1 while Fig. 5(b) presents the final *DAL* after the domination analysis is performed on the three buckets presented in Fig. 4.

Objects		Results of Pairwise Comparison	*DAL*		S_{B_1}	Objects in B_1
			Dominating	*Dominated*		
o_1	o_2	$o_1 \succ o_2$	o_1	$\{o_2\}$	o_1	o_2 is removed from B_1
o_1	o_4	$o_1 \succ o_4$	o_1	$\{o_2, o_4\}$	o_1	o_4 is removed from B_1
o_1	o_{10}	$o_{10} \succ o_1$	o_1	$\{o_2, o_4\}$	o_{10}	o_1 is removed from B_1
			o_{10}	$\{o_1\}$		
o_{10}	o_{12}	$o_{10} \succ o_{12}$	o_1	$\{o_2, o_4\}$	o_{10}	o_{12} is removed from B_1
			o_{10}	$\{o_1, o_{12}\}$		

(a)

DAL	
Dominating	*Dominated*
o_1	$\{o_2, o_4\}$
o_{10}	$\{o_1, o_{12}\}$
o_3	$\{o_8\}$
o_9	$\{o_{15}\}$
o_6	$\{o_5, o_7, o_{11}, o_{12}, o_{14}\}$

(b)

FIGURE 5. **(a) The entries of *DAL* based on B_1 (b) The *Domination Analysis List* (*DAL*).**

Step 3 *Derive the Skyline of Each Bucket:* In this step, the skyline set of each bucket, B_i, is identified. The *Definition* 2 *Skylines* is modified to suit with the scope of bucket as follows:

Definition 14 Skylines of a Bucket: An object $o_i \in B_k$ is a skyline of B_k if there are no other objects $o_j \in B_k$ where $i \neq j$ that dominates o_i.

The domination analysis conducted in the previous step has removed those objects that are dominated by other objects of a bucket. Consequently, the objects left in each bucket are the objects that are not worse than any other objects in the bucket. They are the skyline set of the bucket also known as the bucket skyline. These objects are the best objects within the bucket based on the set of dimensions (criteria) without missing values, d'. For instance, o_{10} is the bucket skyline of bucket B_1 as it is not worse than any other objects in B_1, namely: o_1, o_2, o_4, o_{12} based on $d' = \{d_2, d_3\}$. We use the notation S_{B_i} to represent the set of bucket skylines of bucket B_i. Fig. 6 shows the bucket skylines of B_1, B_2, and B_3 based on the bucket of objects given in Fig. 4. Here, $S_{B_1} = \{o_{10}\}$, $S_{B_2} = \{o_3, o_9\}$, and $S_{B_3} = \{o_6, o_{13}\}$.

Object	d_1	d_2	d_3
o_{10}	–	6	6

S_{B_1} (011)

Object	d_1	d_2	d_3
o_3	7	–	6
o_9	6	–	7

S_{B_2} (101)

Object	d_1	d_2	d_3
o_6	6	4	–
o_{13}	3	6	–

S_{B_3} (110)

FIGURE 6. Example of *Bucket Skylines*.

Step 4: Perform Domination Analysis Between the Bucket Skylines: This step is similar to the second step of *Phase* I. The difference is the domination analysis is performed over the bucket skylines. Every bucket skyline of a bucket is compared to the bucket skylines of other buckets in a pairwise manner. Since each bucket has a unique bitmap representation, it is important to ensure that these objects are comparable on their revised bitwise (See Definition 5 *Revised Bitwise*) before the pairwise comparison is performed. A simple test is performed by performing an AND operation between the bitmap representations of the involved buckets. If it yields 0, then this implies that the objects between these buckets are incomparable; otherwise they are comparable. The results of the domination analysis are captured and appended to the *DAL* explained in Step 2. Besides, for every dominance relationship, say $o_{10} \succ o_6$, identified while performing the domination analysis, the object that dominates other object, o_{10}, is listed in a list called *Dominating Object List* ($\succ OL$) while the object that is dominated, o_6, is listed in a list called *Dominated Object List*($\not\succ OL$). Hence, $\succ OL = \{o_{10}\}$ and $\not\succ OL = \{o_6\}$. If $o_9 \succ o_{10}$, then $\succ OL = \{o_9, o_{10}\}$ and $\not\succ OL = \{o_6, o_{10}\}$. Notice that o_{10} appears in both list. Also, objects that are neither dominates nor being dominated are saved in the $\succ OL$. These lists are significant as the skyline set is determined by comparing the entries of both lists. Fig. 7 presents the results of performing pairwise comparisons between S_{B_1}, S_{B_2}, and S_{B_3} shown in Fig. 6. It also shows the derivation of

Object		Results of Pairwise Comparison	$\succ OL$	$\not\succ OL$
o_{10}	o_3	$o_{10} \not\succ o_3, o_3 \not\succ o_{10}$	o_3, o_{10}	–
o_{10}	o_9	$o_9 \succ o_{10}$	o_3, o_9, o_{10}	o_{10}
o_{10}	o_6	$o_{10} \succ o_6$	o_3, o_9, o_{10}	o_6, o_{10}
o_{10}	o_{13}	$o_{10} \not\succ o_{13}, o_{13} \not\succ o_{10}$	o_3, o_9, o_{10}, o_{13}	o_6, o_{10}
o_3	o_6	$o_3 \succ o_6$	o_3, o_9, o_{10}, o_{13}	o_6, o_{10}
o_3	o_{13}	$o_3 \succ o_{13}$	o_3,o_9, o_{10}, o_{13}	o_6, o_{10}, o_{13}
o_9	o_6	$o_9 \not\succ o_6, o_6 \not\succ o_9$	o_3, o_6, o_9, o_{10}, o_{13}	o_6,o_{10},o_{13}
o_9	o_{13}	$o_9 \succ o_{13}$	o_3, o_6, o_9, o_{10}, o_{13}	o_6, o_{10}, o_{13}

(a)

DAL	
Dominating	*Dominated*
o_1	$\{o_2, o_4\}$
o_{10}	$\{o_1, o_6, o_{12}\}$
o_3	$\{o_6, o_8, o_{13}\}$
o_9	$\{o_{10}, o_{13}, o_{15}\}$
o_6	$\{o_5, o_7, o_{11}, o_{12}, o_{14}\}$

(b)

FIGURE 7. (a) The entries of $\succ OL$ and $\not\succ OL$(b) The updated *DAL*.

the lists $\succ OL$ and $\not\succ OL$ while the shaded row represents the final entries of both lists.

Step 5 *Derive the Final Skylines of* D_I*:* This is the final step of *Phase* I which derives the skyline set, S, of the database, D_I. S is derived by comparing the objects of $\succ OL$ and $\not\succ OL$. There are three cases of memberships with regard to the lists $\succ OL$ and $\not\succ OL$: (i) $o_i \in \succ OL$ and $o_i \in \not\succ OL$ which indicates that the object o_i dominates an object, say o_j, and at the same time it is being dominated by another object say o_k; hence it belongs to both lists. Obviously, o_i is not a skyline. Examples are o_1, o_6 and o_{10}. (ii) $o_i \in \succ OL$ and $o_i \notin \not\succ OL$ which indicates that the object o_i dominates an object, say o_j, and is not being dominated by any other objects; hence o_i is one of the skylines. Examples are o_3 and o_9. (iii) $o_i \notin \succ OL$ and $o_i \in \not\succ OL$ which indicates that the object o_i is dominated by other object say o_j but does not dominate any other objects. Obviously o_i is not a skyline. Thus, the skyline set of D_I. is defined as $S = \{o_i | o_i \in \succ OL \wedge o_i \notin \not\succ OL\}$. Hence, the skyline set of the sample database given in Fig. 3, $S = \{o_3, o_9\}$.

B. PHASE II

In avoiding unnecessary skyline computations when changes are made towards a database owing to a data definition operation(s), we have devised two components that are incorporated into the Δ*Skyline* framework, namely: (i) Δ^+*Skyline* which derives a new skyline set when a new dimension(s) is added to a database and (ii) Δ^-*Skyline* which derives a new skyline set when an existing dimension(s) is removed from a database. Depending on the type of operation, the appropriate component will be invoked. Hence, there is no specific sequence between these components. The following sections deliberate on each of these components in further details.

Δ^+*Skyline*– When a database, D^m, changes its structure and state, due to a new dimension(s) is added, $d_{<add>}$, to D^m, the skyline set, S^m, derived based on D^m is no longer

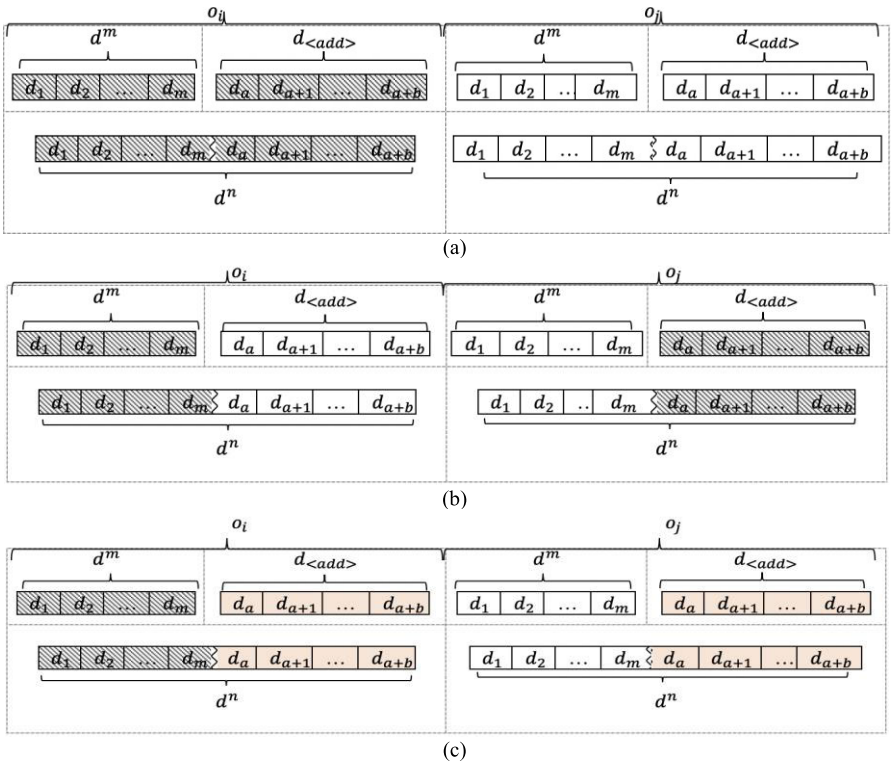

FIGURE 8. Cases between o_i and o_j with $d_{<add>}$.

valid. Intuitively, the new structure and state of the database, denoted as D^n, needs to be analysed in preparing a new skyline set, S^n. Nonetheless, $\Delta^+Skyline$ aims at deriving S^n without the necessity to examine the entire D^n, hence strives to avoid unnecessary skyline computations. Instead, pairwise comparisons are only performed among the affected objects. These objects are those that are dominated by bucket skylines with the domination analysis performed only on the values of the new added dimensions. Our approach is motivated by the following arguments:

Given a database $D^m = \{o_1, o_2, \ldots, o_w\}$ where each object is associated with m dimensions (m criteria) denoted by $d^m = \{d_1, d_2, \ldots, d_m\}$, the skyline set of D^m is S^m which is a set of objects that is not worse than any other objects in D^m which infers that $S^m \subseteq D^m$. When a new dimension(s), $d_{<add>} = \{d_a, d_{a+1}, \ldots, d_{a+b}\}$, is added to D^m, the structure and state of D^m changed which can be represented as follows: $D^n = \{o_1, o_2, \ldots, o_w\}$ with n dimensions (n criteria) denoted by $d^n = \{d_1, d_2, \ldots, d_n\}$, whereby $n = |d^m| + |d_{<add>}|$. The skyline set S^m is no longer valid as the objects that are verified as worse based on d^m might have better, more preferred values with regard to $d_{<add>}$. Since $D^m \subseteq D^n$, conducting domination analysis on the entire D^n would mean repeating the domination analysis on the set of objects $o_1, o_2, \ldots, o_w$ based on d^m, i.e. D^m, besides the D^{n-m} (the new part of D^n). Obviously, this incurs unnecessary skyline computations. Apparently, only the D^{n-m} should be analysed.

Given the $d_{<add>}$, objects that need to be analysed are those that are dominated (worse) by other objects as these objects might have chances to be skylines if their values with regard to $d_{<add>}$ are not worse than the objects that dominate them earlier. Hence, the dominance relationship, $o_i \succ o_j$, captured in the earlier phase needs to be analysed. In our work, the *DAL* as well as the bucket skylines will be utilised. The bucket skylines are objects that are not worse than any other objects in the bucket while the objects that they dominate either within the same bucket or from other buckets are saved in the *DAL*. Hence, given a dominance relationship, $o_i \succ o_j$, where o_i is a bucket skyline while o_j is an object that is dominated by o_i, we have the following $\forall d_k \in d^m, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d^m, o_i.d_l > o_j.d_l$.[2] By analysing o_j against o_i based on $d_{<add>} = \{d_a, d_{a+1}, \ldots, d_{a+b}\}$, the following are observed: (i) o_j is still worse than o_i (ii) o_j is better than o_i, and (iii) o_j might have better values in some of the dimensions of $d_{<add>}$ and worse in other dimensions of $d_{<add>}$. These three cases are elucidated below.

Case I – Totally Dominate: If $\forall d_k \in d_{<add>}, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d_{<add>}, o_i.d_l > o_j.d_l$, then o_i is still a bucket skyline and has a potential to be a skyline of D^n. While o_i is a bucket skyline that dominates o_j based on d^m as shown by the shaded rectangle d^m in Fig. 8(a), o_i is still not worse than o_j based on $d_{<add>}$ as shown by the shaded rectangle $d_{<add>}$ of the figure. This means that o_i is not worse in both sets of dimensions, d^m and $d_{<add>}$, and hence totally dominates o_j as presented by the shaded rectangle d^n of the figure.

[2]Without loss of generality, the *Definition 6 Dominance Relationship on the Revised Bitwise* is referred to where necessary.

Object	d_4	d_5
o_1	9	4
o_2	7	4
o_3	9	4
o_4	4	2
o_5	8	2
o_6	7	9
o_7	6	6
o_8	1	4

Object	d_4	d_5
o_9	8	5
o_{10}	8	5
o_{11}	1	2
o_{12}	4	1
o_{13}	9	5
o_{14}	8	7
o_{15}	4	5

FIGURE 9. Example of D^{n-m} with $d_{<add>}$.

DAL	
Dominating	*Dominated*
o_1	$\{o_2, o_4\}$
o_{10}	$\{o_{12}\}$
o_3	$\{o_8\}$
o_9	$\{o_{10}, o_{15}\}$
o_6	$\{o_7, o_{11}, o_{12}\}$

FIGURE 10. The updated *DAL*.

Case II – Partially Dominate: If $\forall d_k \in d_{<add>}, o_j.d_k \geq o_i.d_k \wedge \exists d_l \in d_{<add>}, o_j.d_l > o_i.d_l$, then o_j which was initially not a bucket syline now has a potential to be a skyline of D^n. While o_i is a bucket skyline that dominates o_j based on d^m as shown by the shaded rectangle d^m in Fig. 8(b), o_i is now worse than o_j based on $d_{<add>}$ as shown by the $d_{<add>}$ of the figure. This means that o_i is not worse than o_j in d^m but worse than o_j in $d_{<add>}$, and hence partially dominates o_j. Similarly, o_j is worse than o_i in d^m but not worse than o_i in $d_{<add>}$, and hence partially dominates o_i. Consequently, both o_i and o_j are said not to dominate each other and are the candidate skylines of D^n.

Case III – Not Dominate: If o_i and o_j do not dominate each other based on $d_{<add>}$, then both o_i and o_j have a potential to be a skyline of D^n. While o_i is a bucket skyline that dominates o_j based on d^m as shown by the shaded rectangle d^m in Fig. 8(c), both o_i and o_j have some values that are better than each other based on $d_{<add>}$. This means that o_i and o_j do not dominate each other on $d_{<add>}$. Consequently, both o_i and o_j are said not to dominate each other and are the candidate skylines of D^n.

In deriving a new skyline set, S^n, the steps described below are followed: (i) Identify the objects that are dominated by a bucket skyline, (ii) perform domination analysis between the objects identified in (i) and the bucket skyline based on $d_{<add>}$, (iii) derive the candidate skylines, (iv) perform domination analysis between the candidate skylines, and (v) derive the final skylines of D^n. Each step is elaborated further in the following paragraphs with $d_{<add>}$ and bucket skylines as given in Fig. 9 and Fig. 6, respectively.

Step 1: Identify the Objects That are Dominated by a Bucket Skyline Given the $d_{<add>}$, for every dominance relationship, $o_i \succ o_j$, that has been established earlier needs to be re-examined as these relationships might no longer be valid. Hence, for each bucket skyline, o_i, the set of objects dominated by o_i is retrieved from the *DAL*. Given the following bucket skylines, $S_{B_1} = \{o_{10}\}$, $S_{B_2} = \{o_3, o_9\}$, and $S_{B_3} = \{o_6, o_{13}\}$, the objects dominated by each of these bucket skylines are retrieved from the *DAL*. For instance, $o_{10} \succ \{o_1, o_6, o_{12}\}$, $o_3 \succ \{o_6, o_8, o_{13}\}$, $o_9 \succ \{o_{10}, o_{13}, o_{15}\}$, $o_6 \succ \{o_5, o_7, o_{11}, o_{12}, o_{14}\}$, while o_{13} does not dominate any objects.

Step 2: Perform Domination Analysis Between the Objects Identified in Step 1 *and the Bucket Skyline Based on* $d_{<add>}$*:* Once the objects that are dominated by each bucket skyline have been identified, domination analysis is then performed between these objects based on the $d_{<add>}$ to update the dominance relationships that have been captured earlier. These updated dominance relationships are saved in the *DAL*. For instance, comparing the values of o_3 against the values of o_6, o_8, and o_{13} based on $d_{<add>} = \{d_4, d_5\}$, the following are observed: (i) o_3 is better than o_6 in d_4 but worse than o_6 in d_5, hence both o_3 and o_6 do not dominate each other with regard to $d_{<add>}$. This example reflects Case III – *Not dominate*. The *DAL* is updated by removing o_6 from the dominated list of o_3. (ii) o_3 is better than o_8 in both dimensions d_4 and d_5. This example reflects Case I – *Totally dominate*. (iii) o_{13} is not worse than o_3 in both dimensions d_4 and d_5, in which o_{13} dominates o_3 with regard to $d_{<add>}$. This example reflects Case II – *Partially dominate*. Here, the *DAL* is updated by removing o_{13} from the dominated list of o_3. For any objects that are no longer dominated like o_1, o_5, o_6, o_{13}, and o_{14}, the dominance relationships of these objects whereby they are the dominating objects (if any) are further analysed. For instance, initially $o_{10} \succ o_1$ however based on $d_{<add>} = \{d_4, d_5\}$, both objects do not dominate each other. Consequently, the dominance relationships $o_1 \succ \{o_2, o_4\}$ captured earlier by *DAL* are re-examined. The same cases as elaborated above are applied. While no further analysis needs to be conducted for the objects o_5, o_6, o_{13}, and o_{14} since they are not listed as the dominating objects in the *DAL*. The updated *DAL* is as shown in Fig. 10. Algorithm 1 presents the domination analysis algorithm for $d_{<add>}$.

Algorithm 1 Domination Analysis Algorithm for $\mathbf{d}_{<\mathbf{add}>}$

Input: $d_{<add>}$; a dominance relationship, $o_i \succ o_j$, where o_i is a bucket skyline and o_j is the object it dominates; *DAL*
Output: Updated *DAL*

```
Begin
    If o_i ≻ o_j on d_<add> Then
        /* Case I – o_i totally dominates o_j */
        Exit
    Else If o_j ≻ o_i on d_<add> Then
        /* Case II – o_i partially dominates o_j
        and vice versa*/
            DAL = DAL − < o_i, o_j >
    Else If o_i ⊁ o_j and o_j ⊁ o_i on d_<add> Then
        /* Case III – o_i and o_j do not dominate
        each other */
            DAL = DAL − < o_i, o_j >
End
```

Step 3: Derive the Candidate Skylines: Based on Step 2, a set of candidate skylines is derived. This is simply obtained by comparing the updated *DAL* against the list of objects a bucket skyline dominates, for instance initially $o_{10} \succ \{o_1, o_6, o_{12}\}$ and based on the updated *DAL*, $o_{10} \succ o_{12}$ which implies that both the objects o_1 and o_6 are no longer being dominated by o_{10} and hence are the candidate skylines. Following the same step, the set of candidate skylines is as follows: $Sc = \{o_1, o_3, o_5, o_6, o_9, o_{10}, o_{13}, o_{14}\}$. Fig. 11 shows the details of the candidate skylines derived based on the given $d_{<add>}$.

Object	d_1	d_2	d_3	d_4	d_5
o_1	–	5	3	9	4
o_3	7	–	6	9	4
o_5	2	3	–	8	2
o_6	6	4	–	7	9
o_9	6	–	7	8	5
o_{10}	–	6	6	8	5
o_{13}	3	6	–	9	5
o_{14}	2	1	–	8	7

FIGURE 11. The candidate skylines after d_4 and d_5 are added.

Step 4: Perform Domination Analysis Between the Candidate Skylines: This step is similar to the Step 4 of *Phase* I. Here, the domination analysis is performed over the candidate skylines, *Sc*. Every candidate skyline is compared to each other in a pairwise manner. Since these candidate skylines might have missing values in different dimensions, hence the AND operation is performed between the bitmap representations of the candidate skylines before comparisons are conducted. The results of the domination analysis are captured and appended to the *DAL*. Besides, for every dominance relationship, say $o_3 \succ o_1$, identified while performing the domination analysis, the object that dominates other object, o_3, is listed in the $\succ OL$ while the object that is dominated, o_1, is listed in the $\not\succ OL$. Also, objects that are neither dominate nor being dominated are saved in the $\succ OL$. Based on the candidate skylines, *Sc*, given in Fig. 11, the $\succ OL = \{o_1, o_3, o_6, o_9, o_{10}, o_{13}, o_{14}\}$ while $\not\succ OL = \{o_1, o_5, o_{10}\}$.

Step 5: *Derive the final skylines of* D^n – This is the final step of *Phase* II which derives the skyline set, S^n, of the database, D^n given the $d_{<add>}$. Similar to the Step 5 of *Phase* I, S^n is derived by comparing the objects of $\succ OL$ and $\not\succ OL$. Thus, the skyline set of D^n is defined as $S^n = \{o_i | o_i \in \succ OL \wedge o_i \notin \not\succ OL\}$. Hence, the skyline set of the sample database given in Fig. 3 with $d_{<add>}$ as shown in Fig. 9, $S^n = \{o_3, o_6, o_9, o_{13}, o_{14}\}$. Comparing S^n to S produced in *Phase* I, i.e. $S = \{o_3, o_9\}$, the objects o_3 and o_9 are still the skylines of the new structure/state of the database, D^n, since they are still not worse than any other objects with regard to the dimensions d_1, d_2, and d_3. This is in line with the cases that we have discussed above. Also, as stated in [21] and [42], the number of skyline results increases exponentially with respect to the size of skyline criteria.

Theorem: For every object, $o_i \in D^n$, not dominated by other objects, $o_j \in D^n$,the Δ^+*Skyline* will identify o_i as a skyline.

Proof: Assume an object $o_i \in D^m$ is a skyline of D^m and after adding $d_{<add>}$ into D^m, $o_i \in D^n$ is not dominated by other objects, $o_j \in D^n$, but $o_i \in D^n$ is not identified as a skyline. An object $o_i \in D^n$ is not a skyline if and only if it is not a bucket skyline or it is a bucket skyline but it is dominated by other bucket skylines. Hence, there are two cases to be considered, as follows:

Case 1: Assume that $o_i \in D^m$ is a bucket skyline, which implies that there is no other objects of the bucket, $o_j \in D^m$, that dominate $o_i \in D^m$. After adding the dimension $d_{<add>}$, assume that $o_i \in D^{n-m}$ is dominated by $o_j \in D^{n-m}$. Based on the *Definition*10 *Dominance Relationship of* D^n*based on* $d_{<add>}$, $o_j \succ o_i$ in which o_i is not a skyline if and only if

1) $\forall d_k \in d^m, o_j.d_k \geq o_i.d_k \wedge \exists d_l \in d^m, o_j.d_l > o_i.d_l$ and $\forall d_k \in d_{<add>}, o_j.d_k \geq o_i.d_k$ or
2) $\forall d_k \in d^m, o_j.d_k \geq o_i.d_k$ and $\forall d_k \in d_{<add>}, o_j.d_k \geq o_i.d_k \wedge \exists d_l \in d_{<add>}, o_j.d_l > o_i.d_l$ hold.

Even though the second condition is true (based on $d_{<add>}$), the first condition is false. Based on Case 1 – Case III of the Δ^+*Skyline*, o_i is a bucket skyline and has a potential to be a skyline, which contradicts the earlier statement. □

Case 2: If $o_i \in D^n$ is a skyline, then $o_i \in$ Bucket Skyline (S_{B_i}) and $o_i \in\!\!> OL$. If o_j dominates o_i, then based on the *Definition 10 Dominance Relationship of* D^n *based on* $d_{<add>}$ $\forall d_k \in d^n, o_j \cdot d_k \geq o_i \cdot d_k \wedge \exists d_l \in d^n, o_j \cdot d_l > o_i \cdot d_l, o_j \in\!\!> OL$ and $o_j \notin\!\!\neq OL$ while $o_i \in\!\!\not> OL$. Since $o_i \in\!\!\not> OL$, o_i is not a skyline, which contradicts the earlier statement. □

From *Case*1 and *Case* 2, the assumption that o_i is a skyline but not identified by the Δ^+*Skyline* is invalid. Hence, the Δ^+*Skyline* derives all the skylines, S^n, of D^n.

Δ^-*Skyline*– When an existing dimension(s), $d_{<remove>}$, is removed from a database, D^m, the skyline set, S^m, derived based on the initial structure and state of D^m is no longer valid. Obviously, a new skyline set, S^n, needs to be derived by examining the new structure and state of the database, denoted as D^n. Nonetheless, in an attempt to avoid unnecessary skyline computations, Δ^-*Skyline* examines only those pairwise comparisons between the affected objects without the necessity to examine the entire D^n. These objects are those that are dominated by bucket skylines with the domination analysis performed only on the values of the dimensions to be removed. Our approach is motivated by the following arguments:

Given a database $D^m = \{o_1, o_2, \ldots, o_w\}$ where each object is associated with m dimensions (m criteria) denoted by $d^m = \{d_1, d_2, \ldots, d_m\}$, the skyline set of D^m is S^m which is a set of objects that is not worse than any other objects in D^m which infers that $S^m \subseteq D^m$. When an existing dimension(s), $d_{<remove>} = \{d_r, d_{r+1}, \ldots, d_{r+s}\}$, is removed from D^m, the structure and state of D^m changed which can be represented as follows: $D^n = \{o_1, o_2, \ldots, o_w\}$ with n dimensions (n criteria) denoted by $d^n = \{d_1, d_2, \ldots, d_n\}$, whereby

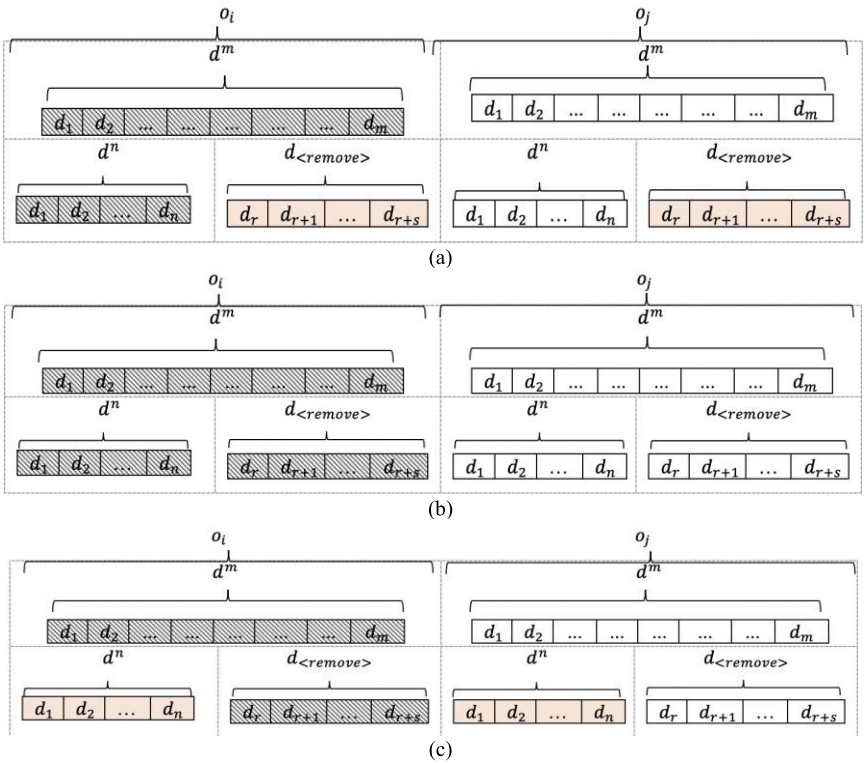

FIGURE 12. Cases between o_i and o_j with $d_{<remove>}$.

$n = |d^m| - |d_{<remove>}|$. The skyline set S^m is no longer valid as the objects that are verified as not worse, say o_i, than other objects, say o_j, based on d^m might no longer have better, more preferred values with regard to d^n. This means that o_i has better value than o_j on the dimension $d_{<remove>}$ which is no longer considered in the skyline computation. Since $D^n \subseteq D^m$, conducting domination analysis on the entire D^n would mean repeating the domination analysis on the set of objects $o_1, o_2, \ldots, o_w$ based on d^n in which $d^n \subset d^m$. Obviously, this incurs unnecessary skyline computations. Apparently, only the D^{m-n} should be analysed.

Given the $d_{<remove>}$, objects that need to be analysed are those that are dominated (worse) by other objects as these objects might have chances to be skylines if their values with regard to d^n are not worse than the objects that dominate them earlier. Hence, the dominance relationship, $o_i \succ o_j$, captured in the earlier phase needs to be analysed. In our work, the *DAL* as well as the bucket skylines will be utilised. The bucket skylines are objects that are not worse than any other objects in the bucket while the objects that they dominate either within the same bucket or from other buckets are saved in the *DAL*. Hence, given a dominance relationship, $o_i \succ o_j$, where o_i is a bucket skyline while o_j is an object that is dominated by o_i, we have the following $\forall d_k \in d^m, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d^m, o_i.d_l > o_j.d_l$.[3] This means that based on d^m, none of the values of o_i is worse than the values of o_j, i.e. either better or equal, while there is at least one value of o_i which is better than the value of o_j, say v_b. Hence, given the $d_{<remove>}$, it is vital to identify whether v_b falls in the $d_{<remove>}$ or d^n. By comparing o_i against o_j based on $d_{<remove>} = \{d_r, d_{r+1}, \ldots, d_{r+s}\}$, the following are observed: (i) the v_b of o_i falls in d^n (ii) the v_b of o_i falls in both $d_{<remove>}$ and d^n, and (iii) the v_b of o_i falls in $d_{<remove>}$. These three cases are elucidated below. For simplicity purposes, we assume that both o_i and o_j are comparable. If otherwise, the *Definition*6 *Dominance Relationship on the Revised Bitwise* is referred.

Case I Dominate on d^n: If $\forall d_k \in d_{<remove>}, o_i.d_k = o_j.d_k$, then o_i is still a bucket skyline and has a potential to be a skyline of D^n. While o_i is a bucket skyline that dominates o_j based on d^m as shown by the shaded rectangle d^m in Fig. 12(a) and the values of o_i are equal to the values of o_j over $d_{<remove>}$, this implies that the v_b of o_i falls in d^n hence removing $d_{<remove>}$ will not affect the dominance relationship between o_i and o_j. This means that o_i is not worse than o_j based on d^n and hence totally dominates o_j as shown by the shaded rectangle d^n in the figure.

Case II Dominate on d^n and $d_{<remove>}$: If $\exists d_l \in d_{<remove>}, o_i.d_l > o_j.d_l$ and $\forall d_k \in d^n, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in d^n, o_i.d_l > o_j.d_l$ hold, then o_i is still a bucket skyline and has a potential to be a skyline of D^n. While o_i is a bucket skyline that dominates o_j based on d^m as shown by the shaded rectangle d^m in Fig. 12(b) and the values of o_i are not worse than the

[3]Without loss of generality, the *Definition*6*Dominance Relationship on the Revised Bitwise* is referred to where necessary.

values of o_j on both the d^n and $d_{<remove>}$, this implies that the v_b of o_i falls in both d^n and $d_{<remove>}$ hence removing $d_{<remove>}$ will not affect the dominance relationship between o_i and o_j. This means that o_i is not worse than o_j based on d^n and hence totally dominates o_j as shown by the shaded rectangle d^n in the figure.

Case III Dominate on $d_{<remove>}$: If $\exists d_l \in d_{<remove>}$, $o_i.d_l > o_j.d_l$ and $\forall d_k \in d^n, o_i.d_k = o_j.d_k$ hold, then o_i is still a bucket skyline and has a potential to be a skyline of D^n. While o_i is a bucket skyline that dominates o_j based on d^m as shown by the shaded rectangle d^m in Fig. 12(c) and the values of o_i are not worse than the values of o_j over $d_{<remove>}$, this implies that the v_b of o_i falls in the $d_{<remove>}$ hence removing $d_{<remove>}$ will affect the dominance relationship between o_i and o_j. Since the values of o_i and o_j are equal over d^n this means that both o_i and o_j do not dominate each other as shown by the rectangle d^n in the figure. Consequently, both o_i and o_j are bucket skylines and both have a potential to be the skylines of D^n.

However, it is not easy to differentiate between cases II and III without analysing the d^n.

In deriving a new skyline set, S^n, similar steps as described in Δ^+*Skyline* are followed. They are: (i) Identify the objects that are dominated by a bucket skyline, (ii) perform domination analysis between the objects identified in (i) and the bucket skyline based on $d_{<remove>}$, (iii) derive the candidate skylines, (iv) perform domination analysis between the candidate skylines, and (v) derive the final skylines of D^n. Each step is elaborated further in the following paragraphs with $d_{<remove>}$ and bucket skylines as given in Fig. 13 and Fig. 6, respectively. Here, we refer to the example given in Fig. 3 in which the dimension d_3 is to be removed.

Object	d_3
o_1	3
o_2	3
o_3	6
o_4	2
o_5	–
o_6	–
o_7	–
o_8	5

Object	d_3
o_9	7
o_{10}	6
o_{11}	–
o_{12}	6
o_{13}	–
o_{14}	–
o_{15}	7

FIGURE 13. Example of D^{m-n} with $d_{<remove>}$.

Step 1: Identify the Objects That Are Dominated by a Bucket Skyline: Given the $d_{<remove>}$, for every dominance relationship, $o_i \succ o_j$, that has been established earlier needs to be re-examined as these relationships might no longer be valid as shown by Case III. Hence, for each bucket skyline, o_i, the set of objects dominated by o_i is retrieved from the *DAL*. Given the following bucket skylines, $S_{B_1} = \{o_{10}\}$, $S_{B_2} = \{o_3, o_9\}$, and $S_{B_3} = \{o_6, o_{13}\}$, the objects dominated by each of these bucket skylines are retrieved from the *DAL*. For instance, $o_{10} \succ \{o_1, o_6, o_{12}\}$, $o_3 \succ \{o_6, o_8, o_{13}\}$, $o_9 \succ \{o_{10}, o_{13}, o_{15}\}$, $o_6 \succ \{o_5, o_7, o_{11}, o_{12}, o_{14}\}$, while o_{13} does not dominate any objects.

DAL	
Dominating	*Dominated*
o_1	$\{o_2, o_4\}$
o_{10}	$\{o_6, o_{12}\}$
o_3	$\{o_6, o_{13}\}$
o_9	$\{o_{13}, o_{15}\}$
o_6	$\{o_5, o_7, o_{11}, o_{12}, o_{14}\}$

FIGURE 14. The updated *DAL*.

Step 2: Perform Domination Analysis Between the Objects Identified in Step 1 *and the Bucket Skyline Based on $d_{<remove>}$:* Once the objects that are dominated by each bucket skyline have been identified, domination analysis is then performed between these objects based on the $d_{<remove>}$ to update the dominance relationships that have been captured earlier. These updated dominance relationships are saved in the *DAL*. For instance, comparing the values of o_{10} against the values of o_1, o_6, and o_{12} based on $d_{<remove>} = \{d_3\}$, the following are observed: (i) o_{10} is better than o_1 in d_3, i.e. $6 > 3$, however whether o_{10} still dominates o_1 can only be verified by analysing the $d^n = \{d_1, d_2\}$. This example reflects Case II and Case III. (ii) o_{10} is not comparable to o_6 on d_3; since o_{10} initially is a bucket skyline while $o_{10} \succ o_6$ based on d^m, then o_{10} must have a value that is better than o_6 in d^n. This example reflects Case I. (iii) o_{10} has the same value as o_{12}, i.e. 6, then o_{10} must have a value that is better than o_{12} in d^n. This example reflects Case I. The updated *DAL* is as shown in Fig. 14. Algorithm 2 presents the domination analysis algorithm for $d_{<remove>}$.

Algorithm 2 Domination Analysis Algorithm for $d_{<remove>}$

Input: $d_{<remove>}$; a dominance relationship, $o_i \succ o_j$, where o_i is a bucket skyline and o_j is the object it dominates; *DAL*
Output: Updated *DAL*

```
1  Begin
2    If o_i ⊁ o_j and o_j ⊁ o_i on d_<remove> Then
         /* Case I – o_i totally dominates o_j */
3          Exit
4    Else If o_i ≻ o_j on d_<remove> and o_i ≻ o_j on d^n Then
         /* Case II – o_i totally dominates o_j^* /
5          Exit
6    Else If o_i ≻ o_j on d_<remove> and o_i ⊁ o_j and o_j ⊁ o_i
     on d^n
      Then /* Case III – o_i and o_j do not dominate
     each other */
7            DAL = DAL − < o_i, o_j >
8  End
```

Step 3: Derive the Candidate Skylines: Based on Step 2, a set of candidate skylines is derived. This is simply obtained by comparing the updated *DAL* against the list of objects a bucket skyline dominates, for instance initially $o_{10} \succ \{o_1, o_6, o_{12}\}$ and based on the updated *DAL*, $o_{10} \succ \{o_6, o_{12}\}$ which implies that the object o_1 is no longer being dominated by o_{10} and hence is one of the candidate skylines. Following the

same step, the set of candidate skylines is as follows: $Sc = \{o_1, o_3, o_6, o_8, o_9, o_{10}, o_{13}\}$. Fig. 15 shows the details of the candidate skylines derived based on the given $d_{<remove>}$.

Object	d_1	d_2
o_1	–	5
o_3	7	–
o_6	6	4
o_8	3	–
o_9	6	–
o_{10}	–	6
o_{13}	3	6

FIGURE 15. The candidate skylines after d_3 is removed.

Step 4: Perform Domination Analysis Between the Candidate Skylines: This step is similar to the Step 4 of *Phase* I. Here, the domination analysis is performed over the candidate skylines, *Sc*. Every candidate skyline is compared to each other in a pairwise manner. Since these candidate skylines might have missing values in different dimensions, hence the AND operation is performed between the bitmap representations of the candidate skylines before comparisons are conducted. The results of the domination analysis are captured and appended to the *DAL*. Besides, for every dominance relationship, say $o_3 \succ o_{13}$, identified while performing the domination analysis, the object that dominates other object, o_3, is listed in the $\succ OL$ while the object that is dominated, o_{13}, is listed in the $\not\succ OL$. Also, objects that are neither dominate nor being dominated are saved in the $\succ OL$. Based on the candidate skylines, *Sc*, given in Fig. 15, the $\succ OL = \{o_1, o_3, o_6, o_9, o_{10}, o_{13}\}$ while $\not\succ OL = \{o_1, o_6, o_8, o_9, o_{13}\}$.

Step 5: Derive the Final Skylines of D^n: This is the final step of *Phase* II which derives the skyline set, S^n, of the database, D^n given the $d_{<remove>}$. Similar to the Step 5 of *Phase* I, S^n is derived by comparing the objects of $\succ OL$ and $\not\succ OL$. Thus, the skyline set of D^n is defined as $S^n = \{o_i | o_i \in \succ OL \wedge o_i \notin \not\succ OL\}$. Hence, the skyline set of the sample database given in Fig. 3 with $d_{<remove>}$ as shown in Fig. 13, $S^n = o_3, o_{10}$. Comparing S^n to S produced in *Phase* I, i.e. $S = \{o_3, o_9\}$, the object o_3 is still the skyline of the new structure/state of the database, D^n, while o_9 is no longer the skyline of D^n.

Theorem: For every object, $o_i \in D^n$, not dominated by other objects, $o_j \in D^n$,the Δ^- *Skyline* will identify o_i as a skyline.

Proof: Assume an object $o_i \in D^m$ is a skyline of D^m and after removing $d_{<remove>}$ from D^m, $o_i \in D^n$ is not dominated by other objects, say $o_j \in D^n$, but $o_i \in D^n$ is not identified as a skyline. An object $o_i \in D^n$ is not a skyline if and only if it is not a bucket skyline or it is a bucket skyline but it is dominated by other bucket skylines. Hence, there are two cases to be considered, as follows:

Case 1: Assume that $o_i \in D^m$ is a bucket skyline, which implies that there is no other objects of the bucket, $o_j \in D^m$, that dominate $o_i \in D^m$. After removing the dimension $d_{<remove>}$, assume that $o_i \in D^{m-n}$ is dominated by $o_j \in D^{m-n}$. Based on the *Definition* 11 *Dominance Relationship of D^nbased on $d_{<remove>}$*, $o_j \succ o_i$ in which o_i is not a skyline if and only if $\forall d_k \in d^n, o_j.d_k \geq o_i.d_k \wedge \exists d_l \in d^n, o_j.d_l > o_i.d_l$ hold. Based on Δ^- *Skyline*, if o_i is a bucket skyline then o_i will remain to be a bucket skyline and has a potential to be a skyline, which contradicts the earlier statement. □

Case 2: If $o_i \in D^n$ is a skyline, then $o_i \in$ *Bucket Skyline* $\left(S_{B_i}\right)$ and $o_i \in\!\succ OL$. If o_j dominates o_i, then based on the *Definition 11 Dominance Relationship of D^n based on* $d_{<\text{remove}>}$, $\forall d_k \in d^n, o_j \cdot d_k \geq o_i \cdot d_k \wedge \exists d_l \in d^n, o_j \cdot d_l > o_i \cdot d_l, o_j \in\!\succ OL$ and $o_j \notin\!\succ OL$ while $o_i \in\!\succ OL$. Since $o_i \in \neq OL$ and based on the Δ^-*skyline*, o_i is not a skyline which contradicts with the earlier statement. □

From *Case*1 and *Case* 2, the assumption that o_i is a skyline but not identified by the Δ^-*Skyline* is invalid. Hence, the Δ^-*Skyline* derives all the skylines, S^n, of D^n.

V. RESULTS AND DISCUSSION

A. EXPERIMENTAL SETTINGS

Several extensive experiments have been designed with the main aim to evaluate the performance of our proposed framework, Δ*Skyline*. As elaborated in the previous sections, Δ*Skyline* is introduced with an attempt to avoid unnecessary skyline computations when changes made towards an incomplete database are due to a new dimension(s) is added to a database or an existing dimension(s) is removed from a database; which will not only change the state of the database but also its structure. The experiments are conducted on Intel Core i7 3.6GHz processor with 32GB of RAM and Windows 8 professional; while VB.NET 2013 is used as the implementation platform of Δ*Skyline*. To verify the arguments made in this paper, we compare the performance of Δ*Skyline* against three notable existing approaches that are designed mainly to deal with the issues related to incompleteness of data, namely: *ISkyline* [24], *SIDS* [6], and *Incoskyline*[3].

Fig. 16 presents the design flow of the experiments. Given an initial incomplete data set with *m* dimensions (criteria) to be considered in the skyline computation, D^m, we first run each algorithm, namely: *ISkyline*, *SIDS*, and *Incoskyline* as well as the *Phase* I of our proposed framework, Δ*Skyline*, to derive the skyline set, S^m. This is represented by Fig. 16(a) and Fig. 16(c), respectively. We use the same notation, S^m, to indicate that each algorithm should produce the same skyline set, S^m, which ensures the correctness of the algorithms. Meanwhile, when changes are made towards, D^m, in which a different set of dimensions (criteria) is to be considered in

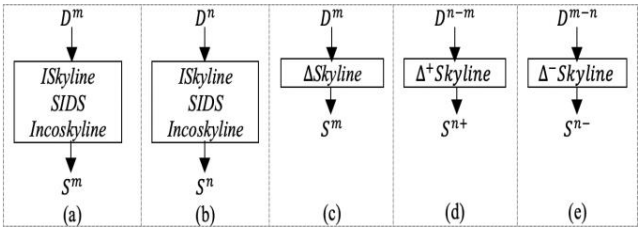

FIGURE 16. The design flow of the experiments.

TABLE 3. The parameter settings of the synthetic and real data sets.

Parameter Settings	Data Sets			
	Synthetic	*NBA*	*stock market*	*MovieLens*
Data Set Size	300K	120K	500K	1200K
Number of Dimensions	15	13	16	4
Incompleteness Rate	20%			
$\|d_{<add>}\|$	1, 2, 4, 6, 8			
$\|d_{<remove>}\|$	1, 2, 4, 6, 8			

TABLE 4. The parameter settings for $\mathbf{d}_{<add>}$ operation.

Data set/Number of dimensions	$\|d^m\|$	$\|d^n\| = \|d^m\| + \|d_{<add>}\|$				
Iteration		**1**	**2**	**3**	**4**	**5**
$\|d_{<add>}\|$		1	2	4	6	8
Synthetic	7	8	9	11	13	15
NBA	5	6	7	9	11	13
stock market	8	9	10	12	14	16
MovieLens	1	2	–	–	–	–

the skyline computation, we run again the *ISkyline*, *SIDS*, and *Incoskyline* on the new structure and state of D^m, denoted as D^n, to derive a new skyline set, S^n. This is as shown in Fig. 16(b). Meanwhile, the Δ*Skyline* is evaluated based on the type of operation used to trigger the changes. If the changes are due to a new dimension(s) is added to the D^m, i.e. $d_{<add>}$, then a new skyline set, S^{n+}, is derived by invoking the Δ^+*Skyline* over the D^{n-m}. Otherwise, a new skyline set, S^{n-}, is derived by invoking the Δ^-*Skyline* over the D^{m-n}. This is represented by Fig. 16(d) and Fig. 16(e), respectively. Note that we use the notations S^{n+} and S^{n-} simply to show that both sets might have different set of objects. However, S^{n+} and S^{n-} should be equal to the set S^n produced by *ISkyline*, *SIDS*, and *Incoskyline* given the same set of $d_{<add>}$ and $d_{<remove>}$, respectively.

Two types of data sets are used in the experiments, namely: synthetic and real data sets. We use the same real data sets as used by the previous works [3], [6], [9], [24], [35], [46], [47], [45] namely: *NBA*, *MovieLens*, and *stock market*. Table 3 presents the parameter settings for both the synthetic and real data sets. The initial sizes of the synthetic, *NBA*, *stock market*, and *MovieLens* data sets are 300K, 120K, 500K, and 1200K, respectively. While, the initial numbers of dimensions for *synthetic*, *NBA*, *stock market*, and *MovieLens* data sets are 15, 13, 16, and 4, respectively. As we regard the data sets as incomplete, hence we set the incompleteness rate to 20% of the size of the data set. Consequently, a data set with the size of 40K will have 4K missing values while a data set with the size of 1200K will have 240K missing values. The number of dimensions to be added, $|d_{<add>}|$, and to be removed, $|d_{<remove>}|$, are varied between 1 and 8 as shown in Table 3.

To ensure the validity of the results, each experiment is run 10 times and the average value of these runs is reported. In deriving the skyline set, we assume that greater values are preferable compared to smaller ones. The performance measurements used in our experiments are number of pairwise comparisons and processing time as they are the most commonly used measurements in evaluating the performance of skyline algorithms [3], [6], [24]. These results are compared to the results of *ISkyline* [24], *SIDS* [6], and *Incoskyline* [3].

B. THE EXPERIMENTAL RESULTS

This section presents the experimental results of Δ*Skyline* in an attempt to avoid unnecessary skyline computations over an incomplete database that changes its structure and state due to a new dimension(s) is added, $d_{<add>}$, to the database or an existing dimension(s), $d_{<remove>}$, is removed from the database.

Effect of Adding a New Dimension(s), $d_{<add>}$: In this section, we illustrate the experimental results of our proposed solution, Δ*Skyline*, and the previous algorithms, namely: *ISkyline* [24], *SIDS* [6], and *Incoskyline* [3], for both the synthetic and real data sets with respect to the number of pairwise comparisons and processing time for the case when changes made to the data sets are due to a new dimension(s) is added, $d_{<add>}$, to the data sets. We start the experiment with the following initial number of dimensions, $|d^m|$: *synthetic* data set with 7, *NBA* data set with 5, *MovieLens* data set with 1, and *stock market* data set with 8 as shown by the column $|d^m|$ of Table 4. Then, a set of dimensions is added to the data sets, $d_{<add>}$, in five iterations with 1, 2, 4, 6, and 8 new dimensions added to the *synthetic*, *NBA*, and *stock market* data sets, while only one new dimension is added to the *MovieLens* data set. This is shown by the column $|d^n| = |d^m| + |d_{<add>}|$ of Table 4. For instance, in iteration 1, one dimension is added to each data set, i.e. $|d_{<add>}| = 1$, which result in $|d^n| = 8$, 6, 9, and 2 dimensions for *synthetic*, *NBA*, *stock market*, and *MovieLens*, respectively.

Figs. 17(a), (b), (c), and (d) present the number of pairwise comparisons achieved by Δ*Skyline*, *ISkyline* [24], *SIDS* [6], and *Incoskyline* [3], based on the *synthetic*, *NBA*, *MovieLens*, and *stock market* data sets, respectively. It is apparent that Δ*Skyline* shows a steady performance for all data sets which reveals that the number of new dimensions added, $|d_{<add>}|$, to the current structure of a data set has no significant impact on the performance of Δ*Skyline*. Intuitively, when the number of added dimensions increases, the number of pairwise comparisons performed increases, as verified in the results of all the previous algorithms. The results of Δ*Skyline* show no exemption, however the increment shown by Δ*Skyline* is small and Δ*Skyline* achieved better performance as compared to *ISkyline*, *SIDS*, and *Incoskyline*, since it avoids unnecessary skyline computations. This is achieved by performing pairwise comparisons only on those affected objects while the dominance relationships between these objects are inferred based on the new added dimensions, $d_{<add>}$. This is realised by utilising the $\succ OL$, $\not\succ OL$, *DAL*, and S_{B_i} that keep track of the dominating objects, dominated objects, dominance relationships, and bucket skylines, respectively. While, *ISkyline*, *SIDS*, and *Incoskyline* perform pairwise

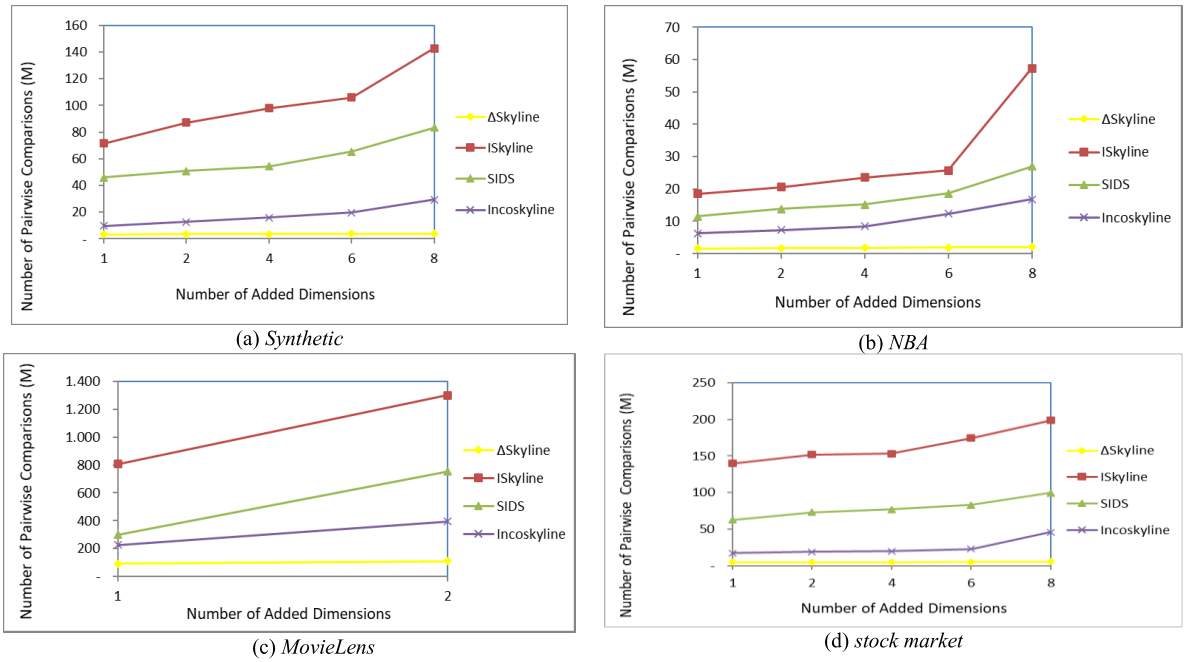

FIGURE 17. The results of number of pairwise comparisons with varying number of new added dimensions.

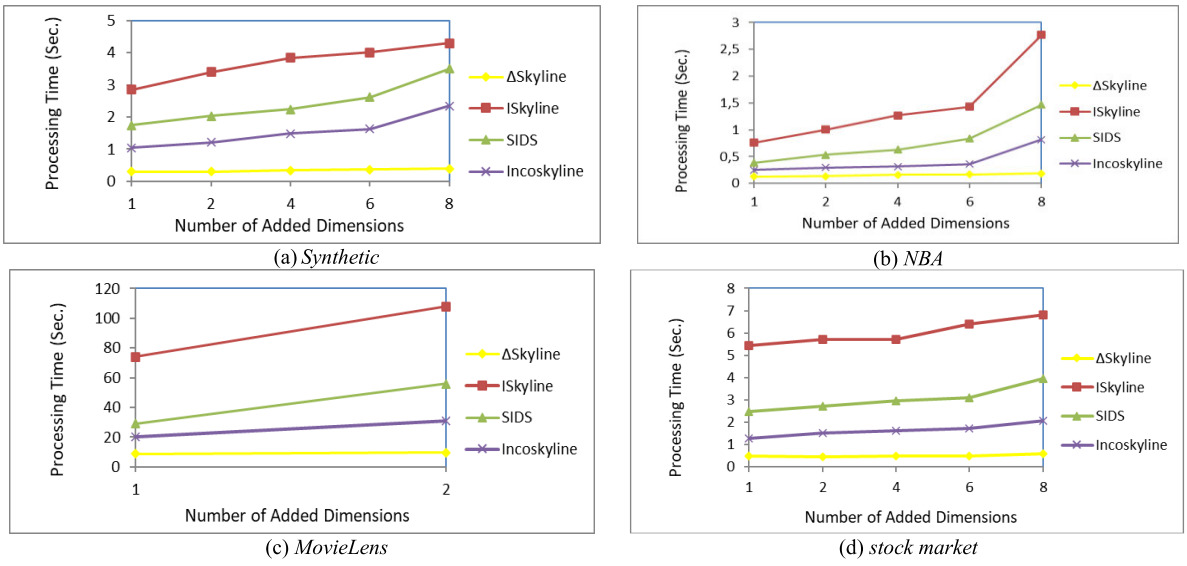

FIGURE 18. The results of processing time with varying number of new added dimensions.

comparisons between all objects by analysing the values of the entire dimensions, i.e. d^n, including those dimensions that are not affected by the changes made towards the data set, i.e. d^m.

Figs. 18(a), (b), (c), and (d) present the processing time achieved by Δ*Skyline*, *ISkyline* [24], *SIDS* [6], and *Incoskyline* [3], based on the *synthetic*, *NBA*, *MovieLens*, and *stock market* data sets, respectively. It is obvious that Δ*Skyline* shows a steady performance which verifies that the number of new dimensions added, $|d_{<add>}|$, to the current structure of a data set has no significant impact on the performance of Δ*Skyline*. Also, Δ *Skyline* achieved less processing time as compared to *ISkyline*, *SIDS*, and *Incoskyline*. Similar trends as presented in Figs. 17(a) – (d) can be seen in Figs. 18(a) – (d). This is due to the fact that reducing the number of pairwise comparisons would reduce the processing time. This is achieved by re-examining only those dominance relationships captured by *DAL* that are affected by the changes over the values of the new added dimensions; hence avoiding performing pairwise comparisons between the objects of the entire data set on all values of the dimensions.

Effect of Removing an Existing Dimension(s) $d_{<remove>}$: In this section, we illustrate the experimental results of our proposed solution, Δ*Skyline*, and the previous algorithms, namely: *ISkyline* [24], *SIDS* [6], and *Incoskyline* [3], for both the synthetic and real data sets with respect to the number of pairwise comparisons and processing time by varying the number of dimensions removed, $d_{<remove>}$, from the existing structure of a data set. We start the experiment with the following initial number of dimensions, $|d^m|$: *synthetic* data set with 15, *NBA* data set with 13, *MovieLens* data set with 4, and *stock market* data set with 16 as shown by the column $|d^m|$ of Table 5. Then, a set of dimensions is removed from the data sets, $d_{<remove>}$, in five iterations with 1, 2, 4, 6, and 8 existing dimensions are removed from the *synthetic*, *NBA*, and *stock market* data sets, while only one new dimension is removed from the *MovieLens* data set. This is shown by the column $|d^n| = |d^m| - |d_{<remove>}|$ of Table 5. For instance, in iteration 1, one dimension is removed from each data set, i.e. $|d_{<remove>}| = 1$, which result in $|d^n| = 14$, 12, 15, and 3 dimensions for *synthetic*, *NBA*, *stock market*, and *MovieLens*, respectively.

TABLE 5. The parameter settings for $d_{<remove>}$ operation.

Data set/Number of dimensions	$\|d^m\|$	$\|d^n\| = \|d^m\| - \|d_{<remove>}\|$				
Iteration		**1**	**2**	**3**	**4**	**5**
$\|d_{<remove>}\|$		1	2	4	6	8
Synthetic	15	14	13	11	9	7
NBA	13	12	11	9	7	5
stock market	16	15	14	12	10	8
MovieLens	4	3	–	–	–	–

Figs. 19(a), (b), (c), and (d) present the number of pairwise comparisons achieved by Δ*Skyline*, *ISkyline* [24], *SIDS* [6], and *Incoskyline* [3], based on the *synthetic*, *NBA*, *MovieLens*, and *stock market* data sets, respectively. As clearly shown in the figures, Δ*Skyline* shows a steady performance which reflects that the number of removed dimensions, $d_{<remove>}$, from the current structure of a data set has no significant impact on the performance of Δ*Skyline*. Intuitively, when the number of removed dimensions increases, the number of pairwise comparisons performed decreases, as reflected in the results of all the previous algorithms. Similar trend can be seen in the results of Δ*Skyline*, however the percentage of decrement shown by Δ*Skyline* is small with better performance as compared to *ISkyline*, *SIDS*, and *Incoskyline*. This is because Δ*Skyline* performs pairwise comparisons only on those affected objects while the dominance relationships between these objects are inferred based on the removed dimensions, $d_{<remove>}$. This is realised by utilising the $\succ OL$, $\not\succ OL$, *DAL*, and S_{B_i} that keep track of the dominating objects, dominated objects, dominance relationships, and bucket skylines, respectively. While, *ISkyline*, *SIDS*, and *Incoskyline* perform pairwise comparisons between all objects by analysing the values of the entire dimensions, d^n, i.e. the dimensions that are not affected by the changes made towards the data set.

TABLE 6. The parameter settings of the synthetic data sets.

Parameter Settings	Data Sets		
	Synthetic (a)	*Synthetic (b)*	*Synthetic (c)*
Data Set Size	120K	300K	500K
Initial number of Dimensions for $d_{<add>}$ analysis	5	7	9
Initial Number of Dimensions for $d_{<remove>}$ analysis	13	15	17
Incompleteness Rate	20%		
$\|d_{<add>}\|$	20%, 40%, 60%, 80%, 100%		
$\|d_{<remove>}\|$	10%, 20%, 30%, 40%, 50%		

Figs. 20(a), (b), (c), and (d) present the processing time achieved by Δ*Skyline*, *ISkyline* [24], *SIDS* [6], and *Incoskyline* [3], based on the *synthetic*, *NBA*, *MovieLens*, and *stock market* data sets, respectively. From these figures, Δ*Skyline* shows a steady performance which reflects that the number of dimensions removed from the current structure of a data set has no significant impact on the performance of Δ*Skyline*. Also, Δ *Skyline* gained less processing time as compared to *ISkyline*, *SIDS*, and *Incoskyline*. Similar trends as presented in Figs. 19 (a) – (d) can be seen in Figs. 20 (a) – (d). This is due to the fact that reducing the number of pairwise comparisons would reduce the processing time. By re-examining only those dominance relationships captured by *DAL* that are affected by the changes over the values of the removed dimensions, the unnecessary pairwise comparisons between the objects are significantly avoided.

Effect of Adding New Dimensions and Removing Existing Dimensions on the Skyline Results and Processing Time: In this section, we illustrate the experimental results of our proposed solution, Δ*Skyline*, on the synthetic data sets with respect to the processing time and the number of skylines produced by *Phase* I denoted as S^m and *Phase* II denoted as S^n. The experiment is conducted over three synthetic data sets with different initial data set sizes as follows: (a) 120K (b) 300K, and (c) 500K while incompleteness rate is set to 20%. The initial numbers of dimensions for each data set are 5, 7, and 9, respectively for the case of adding new dimensions; while 13, 15, and 17, respectively for the case of removing existing dimensions as shown in Table 6. Meanwhile, the number of dimensions added and the number of dimensions removed are based on the percentages of dimensions being added/removed with respect to the total number of dimensions. Here, the round function is used (where necessary) to round the obtained values that contain fractional part to the nearest whole number. The experiment was run 10 times and we report the average value of these runs. These parameter settings are summarised in Table 6.

Intuitively, There Are Two Cases, Namely: (i) $S^m = S^n$ – this indicates that the changes made towards the data set have no effect on the number of skylines produced in *Phase* I as

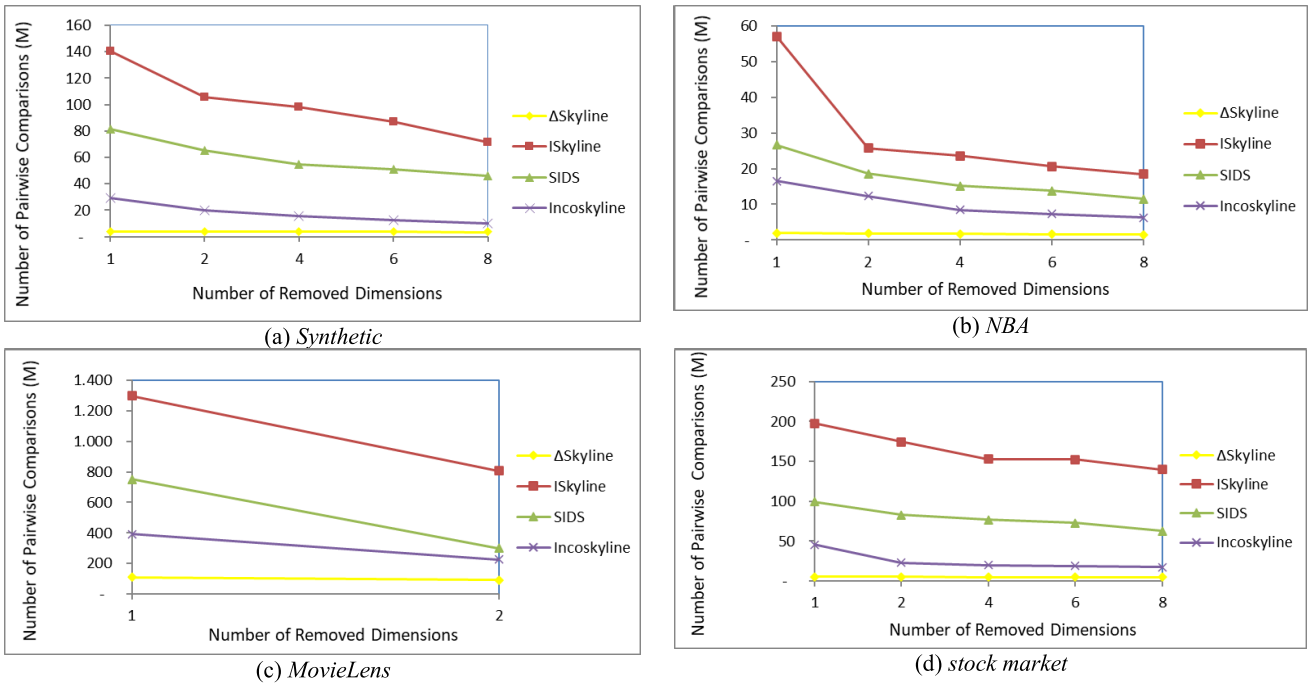

FIGURE 19. The results of number of pairwise comparisons with varying number of removed dimensions.

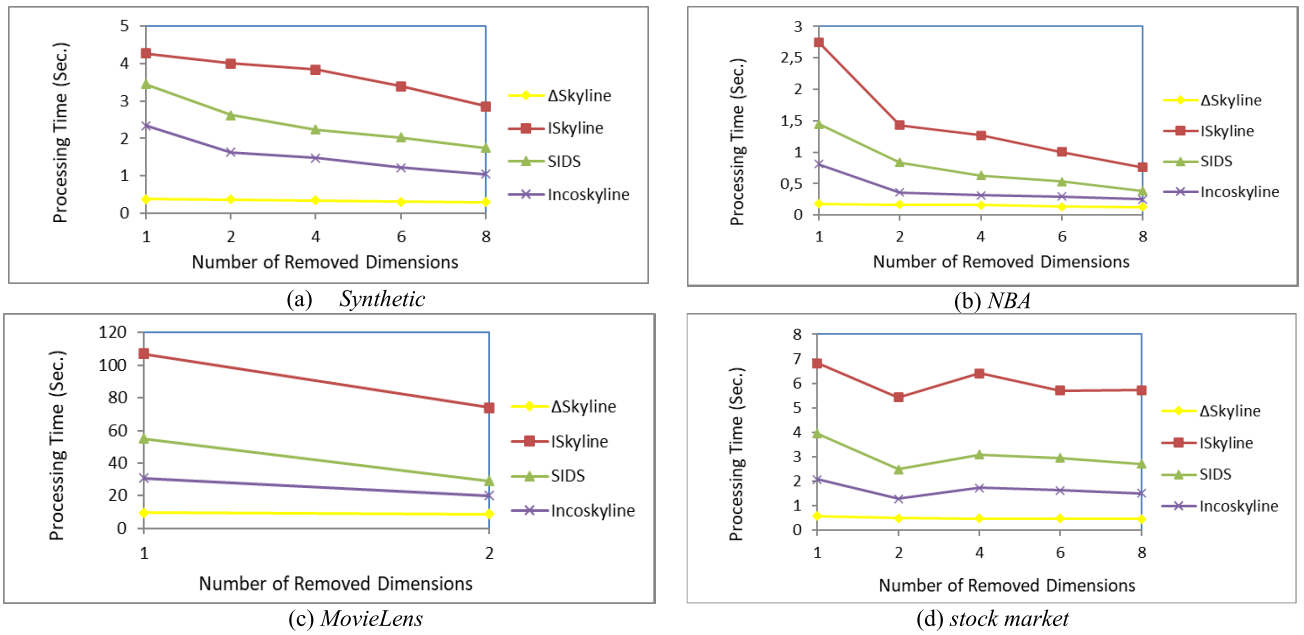

FIGURE 20. The results of processing time with varying number of removed dimensions.

either all the skylines in S^m are also the skylines of S^n or the number of skylines eliminated from S^m (no longer skylines) is equal to the number of new skylines being identified; and (ii) $S^m \neq S^n$ – this case is more realistic and it indicates that the changes made towards the data set affect the skyline set produced in *Phase* I. Fig. 21 shows the results with regard to the number of skylines produced when new dimensions are added, $d_{<add>}$, to the data sets while Fig. 22 presents the results with regard to the number of skylines produced when existing dimensions are removed, $d_{<remove>}$, from the data sets. For all of the runs with $d_{<add>}$ being added to the data sets, the number of skylines produced in *Phase* II is slightly higher than those produced in *Phase* I with all runs showing $S^m \neq S^n$ and $S^m \cap S^n \neq \emptyset$. These results verify

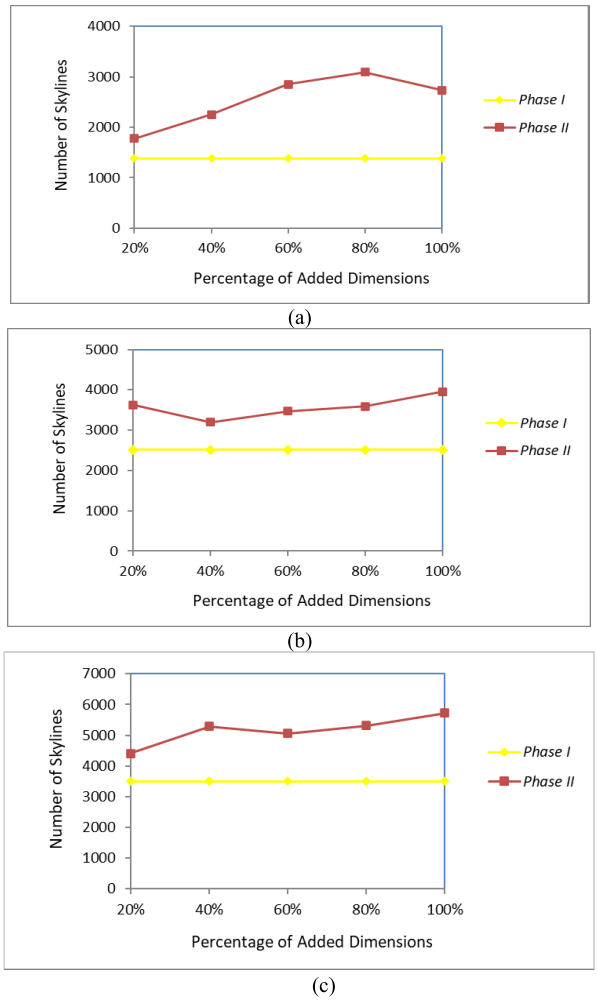

FIGURE 21. The results of number of skylines when new dimensions are added.

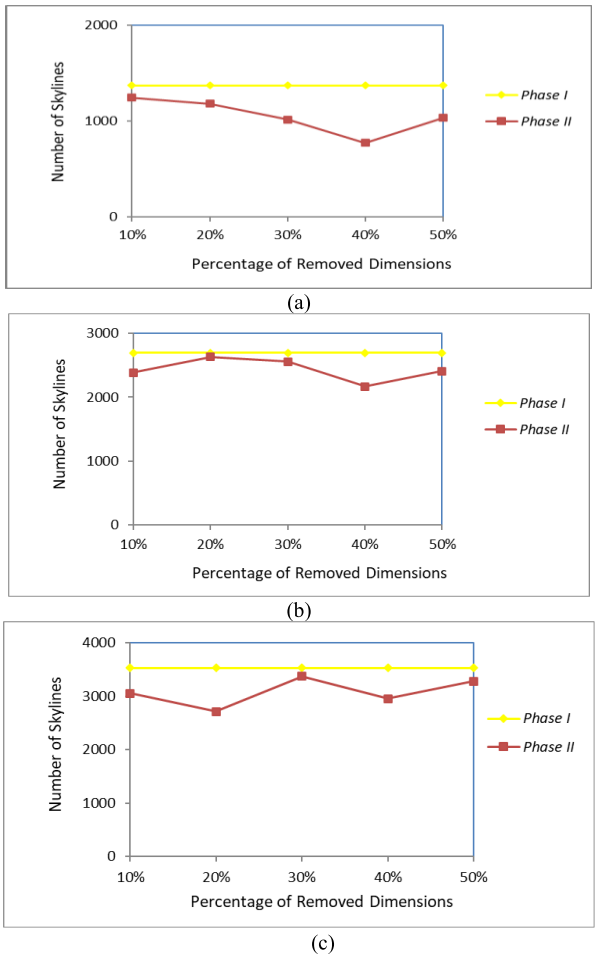

FIGURE 22. The results of number of skylines when existing dimensions are removed.

our cases presented in earlier sections. When a new dimension(s) is added, $d_{<add>}$, for each dominance relationship, $o_i \succ o_j$, o_i is known to be better than o_j on d^m; regardless whether o_i is better or worse than o_j over $d_{<add>}$, o_i is still one of the candidate skylines. While if o_j is better than o_i over $d_{<add>}$, then o_j is one of the candidate skylines which contributes to the increment of the number of skylines. This is also in line with [21], [42], as the number of dimensions (criteria) being considered increases, the number of skylines also increases. Meanwhile, when an existing dimension(s) is removed, $d_{<remove>}$ from the data sets, an opposite trend can be observed. The only case in which the dominance relationship, $o_i \succ o_j$, is affected is when o_i dominates o_j over $d_{<remove>}$. While, for other cases, the dominance relationships are still valid. Nonetheless, the chances for a candidate skyline to be a final skyline is low if the dimension (criteria) being removed is the best value, v_b, that the object has over the other candidate skylines. For instance, consider $o_3(7, –, 6)$ and $o_9(6, –, 7)$. Initially both objects do not dominate each other; however once the third dimension is removed, this results in o_3 dominates o_9. Apparently, the more dimensions are being removed, the more chances that the v_b of an object is being removed.

Fig. 23 shows the results with regard to processing time when new dimensions are added, $d_{<add>}$, to the data sets while Fig. 24 presents the results with regard to processing time when existing dimensions are removed, $d_{<remove>}$, from the data sets. Although the number of skylines produced in *Phase* II is slightly higher than those produced in *Phase* I when new dimensions, $d_{<add>}$, are added to the data sets while opposite trend is observed when the existing dimensions are removed, $d_{<remove>}$ from the data sets as shown in Fig. 21 and Fig. 22 respectively, the processing time taken by *Phase* II for both operations is lower than the processing time taken by *Phase* I for the three synthetic data sets. This is because by re-examining only those dominance relationships captured by *DAL* that are affected by the changes over the values of the added/removed dimensions, the unnecessary pairwise comparisons between the objects are significantly avoided. Consequently, the processing time is significantly reduced.

It is also observed that, as the percentage of added dimensions increases, i.e. the number of dimensions (criteria)

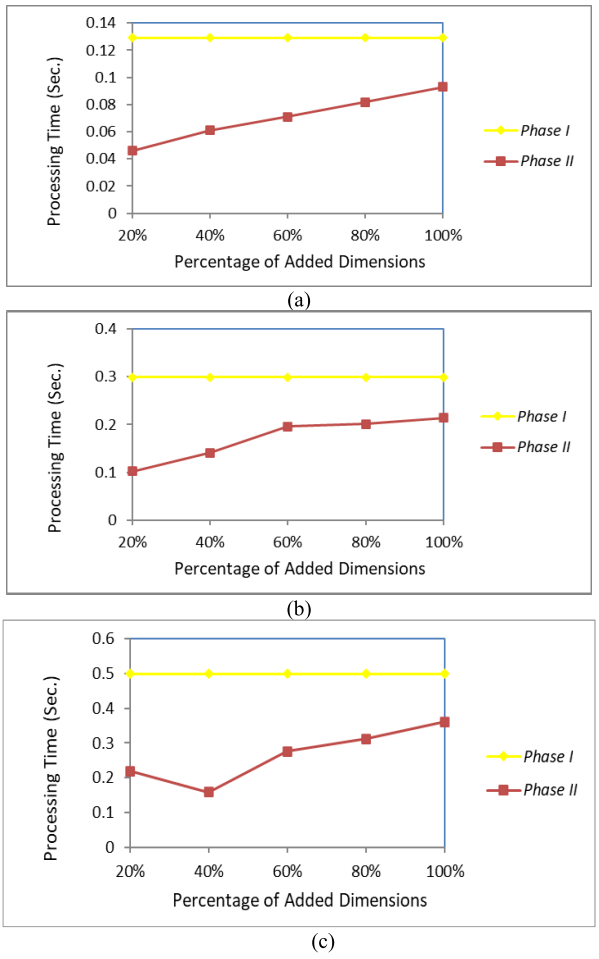

FIGURE 23. The results of processing time when new dimensions are added.

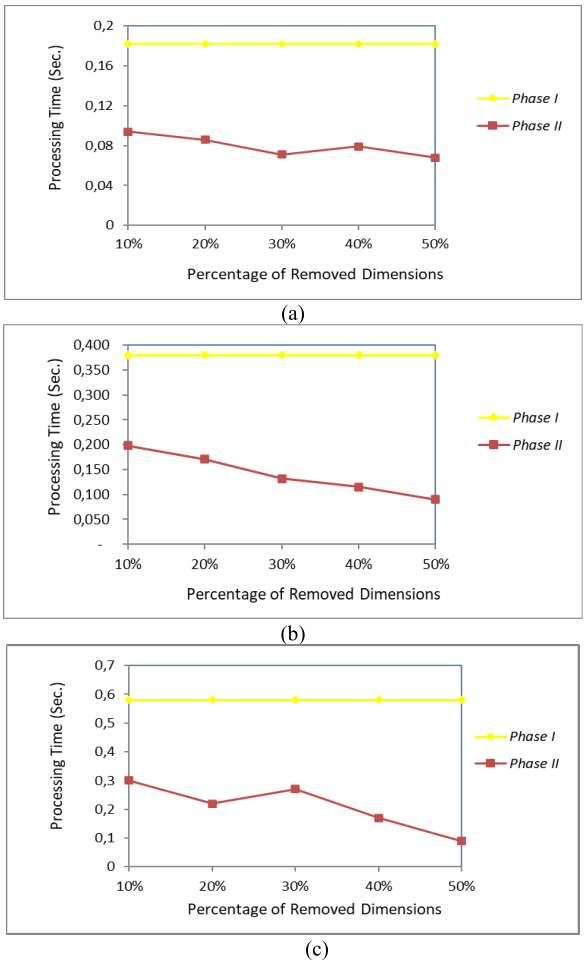

FIGURE 24. The results of processing time when existing dimensions are removed.

considered in the skyline computation increases, the processing time also increases. As explained earlier, as the number of dimensions (criteria) being considered increases, the number of skylines also increases. Consequently, the processing time increases as reflected in Fig. 23. Meanwhile, the opposite trend can be observed when the existing dimensions are removed from the data sets. As the percentage of removed dimensions increases, i.e. the number of dimensions (criteria) considered in the skyline computation decreases, the number of skylines also decreases. As described earlier, the more dimensions are being removed, the more chances that the best value, v_b, of an object is being removed. Consequently, the processing time decreases as reflected in Fig. 24.

Effect of a Sequence of Mixed Operations: In this section, we illustrate the experimental results of our proposed solution, $\Delta Skyline$, on the synthetic data sets with respect to the number of pairwise comparisons and the processing time taken by *Phase* I and *Phase* II for a sequence of mixed operations. The sequence of mixed operations is randomly selected between the add and remove operations. We have designed 4 sets of sequence of mixed operations labelled as 2, 3, 4, and 5 with each indicates the number of operations that are in the sequence while each sequence contains both operations, add and remove. The experiment is conducted over three synthetic data sets with different initial data set sizes as follows: (a) 120K (b) 300K, and (c) 500K while incompleteness rate is set to 20%. The initial numbers of dimensions for each data set are 5, 7, and 9, respectively as shown in Table 7. Meanwhile, the number of dimensions to be added and the number of dimensions to be removed are randomly generated. The experiment was run 10 times and we report the average value of these runs. These parameter settings are summarised in Table 7.

Fig. 25 shows the results of number of pairwise comparisons while Fig. 26 presents the results of processing time for a sequence of mixed operations. Form these figures, it is obvious that for any number of operations in a sequence, the number of pairwise comparisons and processing time of *Phase* II are always lower than *Phase* I. This is due to the fact that for each operation in the sequence of mixed operations, the $\Delta^+ Skyline$ and $\Delta^- Skyline$ utilised the *DAL* to identify the dominance relationships that are affected by the changes made towards the data sets and ignore the necessity

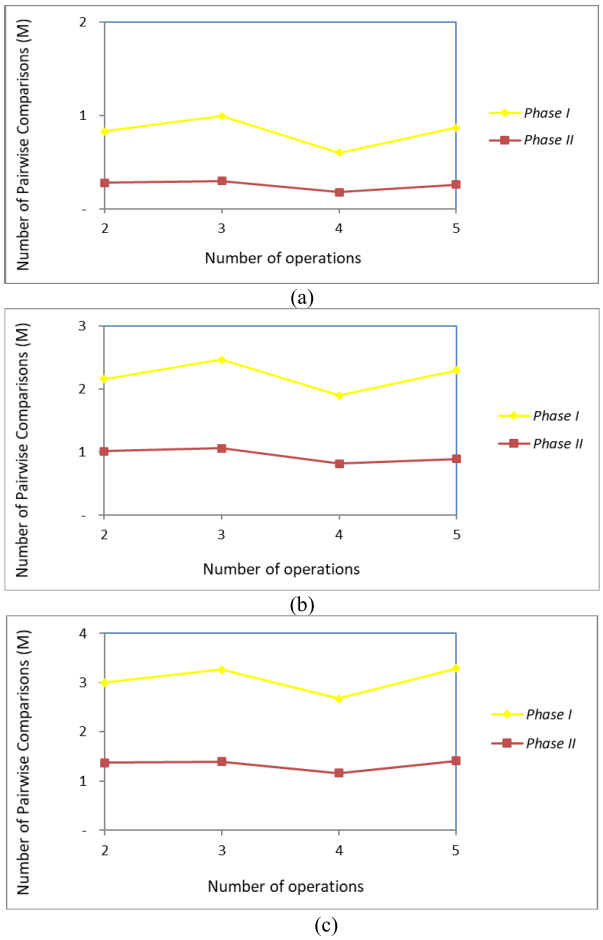

FIGURE 25. The results of number of pairwise comparisons for a sequence of mixed operations.

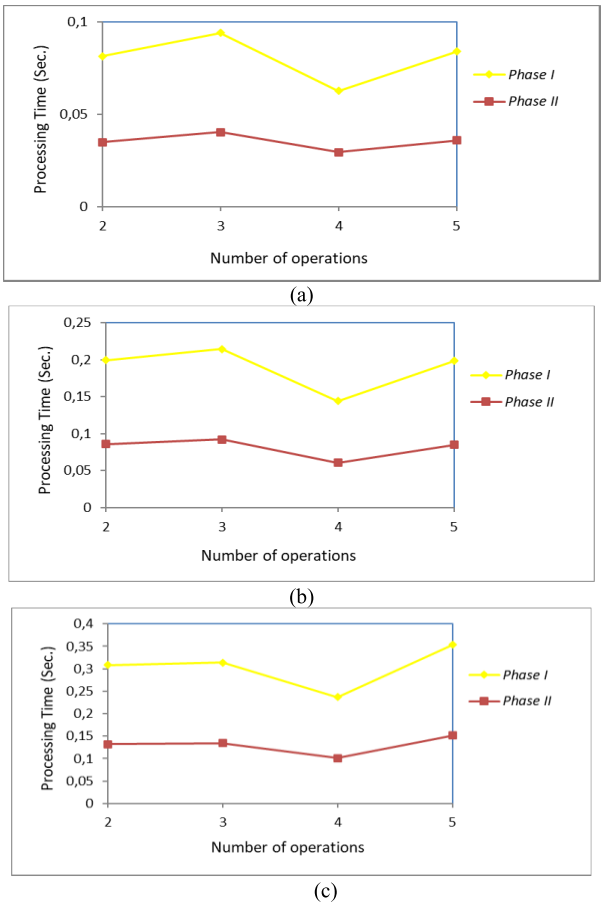

FIGURE 26. The results of processing time for a sequence of mixed operations.

TABLE 7. The parameter settings of the synthetic data sets for a sequence of mixed operations.

Parameter Settings	Data Sets		
	Synthetic (a)	*Synthetic (b)*	*Synthetic (c)*
Data Set Size	120K	300K	500K
Number of Dimensions	5	7	9
Incompleteness Rate	20%		
$\lvert d_{<add>} \rvert$	Randomly generated		
$\lvert d_{<remove>} \rvert$	Randomly generated		

to perform domination analysis on the entire data sets. In this experiment, it is observed that there is no correlation between the number of operations in a sequence and the number of pairwise comparisons; similarly there is no correlation between the number of operations in a sequence and the processing time. Hence, it is unlikely to conclude that as the number of operations in a sequence increases, then the number of pairwise comparisons as well as the processing time also increases as each sequence contains random number of add and remove operations with random number of dimensions added/removed to/from the data sets, that reflects the real-world applications.

VI. CONCLUSION

In this paper, we proposed a solution named $\Delta Skyline$ that is introduced with the main aim to avoid unnecessary skyline computations when a database changes its state and structure due to a data definition operation(s) (add or remove a dimension(s)). $\Delta Skyline$ consists of two optimisation components, namely: $\Delta^{+}Skyline$ which derives a new skyline set when a new dimension(s) is added to a database and $\Delta^{-}Skyline$ which derives a new skyline set when an existing dimension(s) is removed from a database. These components utilise the DAL, $\succ OL$, and $\not\succ OL$ that have been devised to keep track of the domination relationships, dominating, and dominance relationships captured by the DAL; only the necessary pairwise comparisons need to be performed based on the new set of dimensions to be considered in the skyline computation. This has proven to reduce unnecessary skyline computations that are clearly demonstrated in the results of the experiments. Further enhancement to the proposed solution can be done by investigating the following area: data stream, crowd-sourced enabled databases, uncertain database, and big data.

ACKNOWLEDGMENT

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

REFERENCES

[1] K. Alami and S. Maabout, "A framework for multidimensional skyline queries over streaming data," *Data Knowl. Eng.*, vol. 127, May 2020, Art. no. 101792.
[2] 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.
[3] 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.
[4] M. S. Arefin and Y. Morimoto, "Skyline sets queries for incomplete data," *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 5, pp. 67–80, Oct. 2012.
[5] 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.
[6] R. Bharuka and P. S. Kumar, "Finding skylines for incomplete data," in *Proc. 24th Australas. Database Conf.*, vol. 137, 2013, pp. 109–117.
[7] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, Apr. 2001, pp. 421–430.
[8] S. Bothe, P. Karras, and A. Vlachou, "ESkyline: Processing skyline queries over encrypted data," *Proc. VLDB Endowment*, vol. 6, no. 12, pp. 1338–1341, Aug. 2013.
[9] 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.
[10] H. V. Chan, C.-Y. Jagadish, A. K. H. Tan, K.-L. Tung, and Z. Zhang, "On high dimensional skylines," in *Proc. 10th. Int. Conf. Extending Database Technol. (EDBT)*, 2006, pp. 478–495.
[11] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proc. 19th Int. Conf. Data Eng.*, Mar. 2003, pp. 717–719.
[12] X.-W. Cui, L.-G. Dong, H. Zou, and X.-M. An, "Notice of retraction finding k-dominant skyline in dynamic data set," in *Proc. 7th Int. Conf. Natural Comput.*, Jul. 2011, pp. 1247–1250.
[13] 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.
[14] G. B. Dehaki, H. Ibrahim, N. I. Udzir, F. Sidi, and A. A. Alwan, "Efficient skyline processing algorithm over dynamic and incomplete database," in *Proc. 20th Int. Conf. Inf. Integr. Web-Based Appl. Services*, Nov. 2018, pp. 190–199.
[15] Y. Fang and C.-Y. Chan, "Efficient skyline maintenance for streaming data with partially-ordered domains," in *Proc. 15th. Int. Conf. Database Syst. Adv. Appl.*, 2010, pp. 1–16.
[16] X. Fu, X. Miao, J. Xu, and Y. Gao, "Continuous range-based skyline queries in road networks," *World Wide Web*, vol. 20, no. 6, pp. 1443–1467, Nov. 2017.
[17] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *Proc. 31st. Int. Conf. Very Large Data Bases*, 2005, pp. 229–240.
[18] 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.
[19] Y. Gulzar, A. A. Alwan, N. Salleh, I. F. A. Shaikhli, and S. I. M. Alvi, "A framework for evaluating skyline queries over incomplete data," *Procedia Comput. Sci.*, vol. 94, pp. 191–198, Jan. 2016.
[20] Y. Gulzar, A. A. Alwan, and S. Turaev, "Optimizing skyline query processing in incomplete data," *IEEE Access*, vol. 7, pp. 178121–178138, 2019.
[21] X. Han, B. Wang, J. Li, and H. Gao, "Ranking the big sky: Efficient top-k skyline computation on massive data," *Knowl. Inf. Syst.*, vol. 60, no. 1, pp. 415–446, Jul. 2019.
[22] Z. Huang, W. Xu, J. Cheng, and J. Ni, "An efficient algorithm for skyline queries in cloud computing environments," *China Commun.*, vol. 15, no. 10, pp. 182–193, Oct. 2018.
[23] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for uncertain data," in *Proc. Conf. Inf. Knowl. Manage.*, 2010, pp. 1293–1296.
[24] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for incomplete data," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Apr. 2008, pp. 556–565.
[25] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos, "Continuous processing of preference queries in data streams," in *Proc. 36th. Int. Conf. Current Trends Theory Pract. Comput. Sci.*, 2010, pp. 47–60.
[26] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proc. 28th. Int. Conf. Very Large Data Bases*, 2002, pp. 275–286.
[27] M. M. Lawal, H. Ibrahim, N. F. M. Sani, and R. Yaakob, "An indexed non-probability skyline query processing framework for uncertain data," in *Proc. Int. Conf. Adv. Mach. Learn. Technol. Appl.*, 2020, pp. 289–302.
[28] J. Lee, H. Im, and G.-W. You, "Optimizing skyline queries over incomplete data," *Inf. Sci.*, vols. 361–362, pp. 14–28, Sep. 2016.
[29] X. Lian and L. Chen, "Monochromatic and bichromatic reverse skyline search over uncertain databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2008, pp. 213–226.
[30] L. Libkin, "Incomplete data: What went wrong, and how to fix it," in *Proc. 33rd ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, Jun. 2014, pp. 1–13.
[31] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure and efficient skyline queries on encrypted data," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 7, pp. 1397–1411, Jul. 2019.
[32] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 633–644.
[33] 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.
[34] X. Miao, Y. Gao, G. Chen, and T. Zhang, "K-dominant skyline queries on incomplete data," *Inf. Sci.*, vols. 367–368, pp. 990–1011, Nov. 2016.
[35] M. Morse, J. M. Patel, and W. I. Grosky, "Efficient continuous skyline computation," *Inf. Sci.*, vol. 177, no. 17, pp. 3411–3437, Sep. 2007.
[36] W. Nutt, S. Razniewski, and G. Vegliach, "Incomplete databases: Missing records and missing values," in *Proc. 17th. Int. Conf. Database Syst. Adv. Appl.*, 2012, pp. 298–310.
[37] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 41–82, Mar. 2005.
[38] 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.
[39] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proc. Int. Conf. Very Large Database*, 2007, pp. 15–26.
[40] Y. Qi and M. J. Atallah, "Identifying interesting instances for probabilistic skylines," in *Proc. Int. Conf. Database Expert Syst. Appl.*, 2010, pp. 300–314.
[41] 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. 13th. Int. Baltic Conf. Databases Inf. Syst.*, 2018, pp. 272–286.
[42] T. V. Saradhi, K. Subrahmanyam, P. V. Rao, and H.-J. Kim, "Applying Z-curve technique to compute skyline set in multi criteria decision making system," *Int. J. Database Theory Appl.*, vol. 9, no. 12, pp. 9–22, Dec. 2016.
[43] M. A. Soliman, I. F. Ilyas, and S. Ben-David, "Supporting ranking queries on uncertain and incomplete data," *VLDB J.*, vol. 19, no. 4, pp. 477–501, Aug. 2010.
[44] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *Proc. 27th. Int. Conf. Very Large Data Bases*, 2001, pp. 301–310.
[45] G. Wolf, A. Kalavagattu, H. Khatri, R. Balakrishnan, B. Chokshi, J. Fan, Y. Chen, and S. Kambhampati, "Query processing over incomplete autonomous databases: Query rewriting using learned data dependencies," *VLDB J.*, vol. 18, no. 5, pp. 1167–1190, Oct. 2009.
[46] R. C. W. Wong, A. W. C. Fu, J. Pei, Y. S. Ho, T. Wong, and Y. Liu, "Efficient skyline querying with variable user preferences on nominal attributes," in *Proc. 34th. Int. Conf. Very Large Data Bases (VLDB)*, 2008, pp. 1032–1043.
[47] P. Wu, D. Agrawal, O. Egecioglu, and A. el Abbadi, "DeltaSky: Optimal maintenance of skyline deletions without exclusive dominance region generation," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 486–495.
[48] M. L. Yiu and N. Mamoulis, "Multi-dimensional top-k dominating queries," *VLDB J.*, vol. 18, no. 3, pp. 695–718, Jun. 2009.

[49] K. Zhang, H. Gao, X. Han, Z. Cai, and J. Li, "Probabilistic skyline on incomplete data," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 427–436.
[50] K. Zhang, H. Gao, X. Han, Z. Cai, and J. Li, "ISSA: Efficient skyline computation for incomplete data," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2016, pp. 321–328.
[51] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu, "Probabilistic skyline operator over sliding windows," in *Proc. IEEE 25th Int. Conf. Data Eng.*, Mar. 2009, pp. 1060–1071.

GHAZALEH BABANEJAD DEHAKI was born in Tehran, Iran, in March 1981. She received the bachelor's degree in the field of software engineering from the Faculty of Computer Science, Iran University of Science and Technology, in 2007, the master's degree in knowledge management with multimedia from Multimedia University (MMU), Malaysia, specializing in semantic web ontology and recommender systems, and the Ph.D. degree from Universiti Putra Malaysia (UPM) in the field of database systems, specializing in preference query in dynamic and incomplete database systems.

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 with 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, and concurrency control; and data management in mobile, grid, and cloud.

ALI A. ALWAN received the master's and Ph.D. degrees in computer science from Universiti Putra Malaysia (UPM), Malaysia, in 2009 and 2013, respectively. He is currently an Associate Professor with the Kulliyyah (Faculty) of Information and Communication Technology, International Islamic University Malaysia (IIUM), Malaysia. His research interests include preference queries, skyline queries, probabilistic and uncertain databases, query processing and optimization and management of incomplete data, data integration, location-based social networks (LBSNs), recommendation systems, and data management in cloud computing.

FATIMAH SIDI received the Ph.D. degree in management information system from Universiti Putra Malaysia (UPM), Malaysia, in 2008. She is currently working as an Associate Professor with the Discipline of Computer Science, 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, and database and data warehouse.

NUR IZURA UDZIR received the Bachelor of Computer Science and Master of Science degrees from 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 with 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.