IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 23, 2020, accepted June 2, 2020, date of publication June 8, 2020, date of current version June 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3000664

Skyline Queries Computation on Crowdsourced-Enabled Incomplete Database

MARWA B. SWIDAN[1], ALI A. ALWAN[1], SHERZOD TURAEV[2], HAMIDAH IBRAHIM[3], ABEDALLAH ZAID ABUALKISHIK[4], AND YONIS GULZAR[5]

[1]Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Kuala Lumpur 53100, Malaysia
[2]Department of Computer Science and Software Engineering, College of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates
[3]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia
[4]College of Computer Information Technology, American University in the Emirates, Dubai, United Arab Emirates
[5]Department of Management Information Systems, College of Business Administration, King Faisal University, Al-Ahsa 31982, Saudi Arabia

Corresponding authors: Ali A. Alwan (aliamer@iium.edu.my) and Sherzod Turaev (sherzod@uaeu.ac.ae)

This work was supported in part by the Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Malaysia. The work of Sherzod Turaev was supported by the United Arab Emirates University Start-Up under Grant G00003321.

ABSTRACT Data incompleteness becomes a frequent phenomenon in a large number of contemporary database applications such as web autonomous databases, big data, and crowd-sourced databases. Processing skyline queries over incomplete databases impose a number of challenges that negatively influence processing the skyline queries. Most importantly, the skylines derived from incomplete databases are also incomplete in which some values are missing. Retrieving skylines with missing values is undesirable, particularly, for recommendation and decision-making systems. Furthermore, running skyline queries on a database with incomplete data raises a number of issues influence processing skyline queries such as losing the *transitivity property* of the skyline technique and *cyclic dominance* between the tuples. The issue of estimating the missing values of skylines has been discussed and examined in the database literature. Most recently, several studies have suggested exploiting the crowd-sourced databases in order to estimate the missing values by generating plausible values using the crowd. Crowd-sourced databases have proved to be a powerful solution to perform user-given tasks by integrating human intelligence and experience to process the tasks. However, task processing using crowd-sourced incurs additional monetary cost and increases the time latency. Also, it is not always possible to produce a satisfactory result that meets the user's preferences. This paper proposes an approach for estimating the missing values of the skylines by first exploiting the available data and utilizes the implicit relationships between the attributes in order to impute the missing values of the skylines. This process aims at reducing the number of values to be estimated using the crowd when local estimation is inappropriate. Intensive experiments on both synthetic and real datasets have been accomplished. The experimental results have proven that the proposed approach for estimating the missing values of the skylines over crowd-sourced enabled incomplete databases is scalable and outperforms the other existing approaches.

INDEX TERMS Incomplete data, crowdsourcing databases, approximate functional dependencies, query processing, skylines, skyline queries.

I. INTRODUCTION

Nowadays, the issue of missing data has become a frequent phenomenon in many non-trivial numbers of database applications [1], [2]. There are many causes for data incompleteness in database, this includes but not limited to automatic information extraction or information integration, data integration, incomplete input by ignoring some data. These activities negatively influence the database contents and deteriorate their quality [1]–[3]. These factors impact on the completeness and the correctness of the query result [4]–[7]. Some queries cannot be optimally answered through traditional database management techniques as the process of answering certain queries relies on information that is incomplete, imprecise, or uncertain. Examples of such problems include translation, handwriting recognition, image

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

understanding, and web databases. Crowdsourcing has become an effective solution for such types of queries by exploiting knowledge, ideas, experiences, and skills of crowd workers to process information and obtain accurate answers to difficult or very cost-intensive queries. The process of integrating individuals who carry out computations using software systems is known as hyped human/machine systems [8]–[11]. Several crowd-sourcing database systems have already been developed to extend the traditional databases system into crowd-sourced databases systems. This extension supports more types of queries by means of the power of people such as CrowdDB [8], Qurk [12], Deco [13].

In recent years, skyline queries have gained considerable attention for their usefulness in assisting in multi-criteria decision making and decision support applications. The main aim of skyline queries is to generate the best tuples (skylines) for the users based on their preferences. Skyline queries have been investigated by enormous number of researchers since its first introduction into the database community by Borzsony in 2001 [14]. Numerous preference evaluation algorithms that apply the skyline concept have already been suggested in the literature such as Divide-and-Conquer (D&C), Block Nested-Loop (BNL) [14], Sort Filter Skyline (SFS) [15], Branch and Bound Skyline (BBS) [16], Linear Elimination Sort Skyline (LESS) [17], ISKY and NSKY [18], and a geometry-based distributed spatial query strategy (GDSSky) [19]. Most of these previous approaches designed for computing skylines attempt to reduce the search space to avoid the exhaustive searching during the skyline operation. This limits the focus to tuples that have high potential to be in the skyline result over a database with complete data (no missing values). The searching space is determined by the number of pairwise comparisons that need to be performed between the tuples in identifying skylines. That means a higher number of pairwise comparisons results into a larger searching space and vice versa.

Data incompleteness has a severe impact on skyline queries computation. Among the critical issue of data incompleteness in skyline queries is that the *transitivity property* of the skyline technique is lost and that the dominance relationship between tuples becomes *cyclic*. These two problems occurred due to the fact that some tuples are incomparable with each other and thus no tuple is considered as skyline [20]. To clarify these two problems, the following incomplete database example is given. Assume a database consists of three tuples with missing values in one of more attributes, namely a(4, 2, ?), b(3, ?, 4), and c(?, 3, 2). Here the missing values have been replaced with (?). Applying the skyline technique on the three tuples with missing values results into the following: First, tuple a dominates b based on the common non-missing attributes (A_1), and b dominates c on the common non-missing attribute (A_3). Thus, it can be observed that tuple c dominates a based on the common non-missing attribute (A_2). Therefore, the *transitivity property* of the skyline technique no longer holds and the dominance relationship is *cyclic* as none of these three tuples can be considered a skyline as each tuple is dominated by at least one other tuple [20]–[24].

To solve the above problem of skyline computation over incomplete data, many approaches of skyline query processing on the traditional incomplete database have been proposed. This includes Iskyline [20], RBSSQ [25], Sort-based Incomplete Data Skyline (SIDS) [26], Incomplete Data Frequent Skyline (IDFS) [27], KISB and VP [28], IncoSkyline [21], IDS and SPQ [29], ISSA [30] and SCSA [22]. These approaches have introduced solutions to the issue of processing skyline queries on incomplete databases without paying attention to manipulating the missing values of the skylines. Apart for the impact of data incompleteness on skyline query computation, the skylines produced from the incomplete database are also incomplete in which some values of skylines in one or more attributes are missing (not present). Retrieving skylines with incomplete data is thus considered a dissatisfactory approach as it may lead to undesirable results that misleading the users and ends with inappropriate selection and wrong decision. Therefore, manipulating these incomplete skylines by replacing the missing values with some plausible values is desirable as it provides the users with complete skylines that help them to make the best decision. This paper attempts to investigate the impact of processing skyline queries on crowd-sourced enabled incomplete databases. The focus is directed towards how to generate precise estimated values for the missing values of the skylines by exploiting the available data in the incomplete database and benefit from the embedded relationship between the attributes and the crowd-sourced databases.

To the best of our knowledge, the only work that has so far raised the issue of value estimation for skylines in the traditional database is introduced in [31]. His research team proposed an approach that attempts to manipulate the missing values of the skylines utilizing the concept of approximate functional dependencies and the probability correlations between the attributes to estimate the missing values of the skylines. Therefore, the skylines of the database with incomplete data return to the user with complete data. Nevertheless, it is not always suitable to use the available data in the incomplete database to estimate the missing values. In certain cases, the relative error produced between the real missing and the estimated values can be very large and deteriorate the quality of skylines. Thus, outsourcing the missing values using a crowd-sourced database provide a more precise estimation.

Skyline queries are expensive operations, especially when executed on the crowd-sourced enabled databases. This is because the crowd contains a large amount of data for various type of databases, and interfering humans to fill those missing values involves additional monetary cost and results in latency as all missing values need to be elicited from the crowd-sourcing database.

To the best of our knowledge, not sufficient attention has been paid to processing skyline queries in the crowd-sourcing enabled and incomplete database. So far only two approaches have been proposed. The first approach proposed by the work

presented in [5] consists of an innovative hybrid approach that combines crowd-sourcing with heuristic techniques for computing skylines on incomplete data with maximum result quality and acceptable costs. The heuristic technique on which their work mainly relies on k-nearest neighbour imputation (KNN). In 2013 the work in [6] proposed a new approach aiming to overcome the limitation of the previous work by proposing a different heuristic approach than KNN. The new approach adopts a model named minimum value model and has been further extended in 2015 [4] by adding the two new strategies of crowd-sourcing and advanced heuristics. The crowd-sourcing strategy is based on incorporating human workers to attain improved results, while the advanced heuristic offers an alternative offline solution for times when crowd-sourcing may not be a feasible option, for example when the missing data are not easily available for the crowd or the costs of crowd-sourcing are prohibitive. We conclude that the approach introduced in [4] has failed to generate an accurate value when the missing rate is very high. Besides, the approach incurred high time latency and monetary cost when estimating the missing values from the crowd. Most importantly, the process of skyline queries has been performed after estimating the missing values and there are many unnecessary tuples which are non-skylines have been processed. Thus, processing these dominated tuples can be avoided if a data filtration process is applied before estimating the missing values. Hence, a significant amount of time and cost can be saved.

The main contribution of this paper has been summarized in the following points.

- We discuss the issue of skyline query computation over crowd-sourced enabled incomplete databases and explain the need for a new solution to provide precise estimation for the missing values of skylines.
- We conduct a comprehensive survey for the most well-known studies conducted on processing skyline queries in databases systems. This encompasses the previous approaches proposed in the context of traditional incomplete databases and crowd-sourced enabled incomplete databases.
- We propose an efficient filtration strategy that aims at prunes the unwanted dominated tuples from the initial incomplete databases before applying the process of value estimation for the missing values of the skylines.
- We propose an approach named AFD-based that produces precise estimations for the missing skyline values by exploiting the available data in the initial incomplete database and the implicit relationships between the attributes. Exploiting the initial incomplete data and the implicit relationships between the attribute allows to simplify the process of value estimation and minimize the relative error between the real and the missing values of the skylines. The proposed plan results in minimum monetary cost and time latency while maintaining high precision for the estimated values of the skylines using the crowd-sourced databases.
- We evaluate the efficiency and the effectiveness of the proposed solution through several experiments using both real and synthetic datasets.

This paper is organized as follows. In Section II, the previous works related to the area of processing skyline queries over the traditional incomplete databases and the crowd-sourced enabled incomplete databases are presented and discussed. The necessary definitions and notations, which are used throughout the paper, are described in Section III. Section IV elaborates our proposed approach for processing skyline queries in crowd-sourced enabled incomplete databases which has seven main phases. A running database example is also given to clarify the phases. The experimental result is demonstrated in Section V. Conclusion and further research direction are depicted in the final Section VI.

II. RELATED WORK

This section explains and examines the previous approaches designed for processing skyline queries over a database with incomplete data. The section covers examining the techniques proposed for skyline queries computation on traditional incomplete databases and the crowd-sourced enabled incomplete databases.

A. SKYLINE QUERIES FOR INCOMPLETE DATABASES

Several approaches have been proposed for handling missing information in skyline computation [21]–[28], [36], [37]. This section presents previous approaches proposed to process skyline queries in the incomplete traditional database.

The work contributed by [20] is the first attempt to resolve the issue of skyline queries in a database with partial incomplete data. They proposed two algorithms for handling the skyline queries in incomplete data, namely *Bucket* and *Iskyline*. The *Bucket* algorithm divides the tuples of the database into distinct buckets based on their bitmap representation. Each bucket contains the tuples which have missing values in the same attributes. The conventional skyline algorithm is utilized on each bucket to identify the local skylines. Finally, the local skylines of each bucket are compared to each other to derive the final skylines. The *Iskyline* algorithm handles the skyline queries in the incomplete traditional database by dividing the initial database into distinct nodes depending on the missing values of attributes before applying the conventional skyline technique to retrieve the local skylines in every node. The *Iskyline* method conducts two optimization techniques that reduce the number of local skylines in every node. However, *Iskyline* is rather time-consuming as in each node there are many pairwise comparisons that need to be performed to find the local skylines.

In 2012 Arefin and Morimoto proposed the *Replacement-Based Sets Skyline Computation* algorithm (RBSSQ) based on the skyline sets queries from databases with incomplete data [25]. It uses a replacement-based approach and is applicable to the databases having any number of missing attributes in the database tuples. This algorithm consists of

two phases: data pre-processing and skyline sets computation. In the data pre-processing phase, the incomplete values from each attribute are found, then the incomplete values of the tuples in an attribute are replaced with a value outside the domain values. The choice of such a value for an attribute depends on the nature of the data in that attribute. In the Skyline sets, computation phase skyline sets are computed from the processed data. However, experimental evaluation demonstrates that the proposed algorithm is meaningful and scalable enough to handle large and high dimensional data sets.

The *Sort-based Incomplete Data Skyline* algorithm (SIDS) has been developed to compute the skylines over incomplete data as proposed by [26]. The proposed approach chooses one of the attributes following the round-robin method, and the tuple with the next best value in that attribute being chosen for processing. The aim is to process relatively dominant tuples early so that non-skyline tuples can be pruned as early as possible. Consequently, fewer comparisons are required to determine the skyline, which reduces the execution time of the algorithm. The algorithm initially considers all tuples in the data set as candidate skylines before removing dominated tuples from the candidate set. If a tuple has not been pruned yet and has been processed k times, where k is the count of complete attributes for the tuple, then it is determined to be a skyline and can be returned immediately. The rationale here is that any tuple with k complete attributes can be dominated in at most k attributes. Thus, their algorithm can progressively return skylines whenever the above condition is satisfied. However, the proposed approach has been designed to work on traditional databases, which differs completely from the crowd-sourcing database and results in a large number of pairwise comparisons to identify skylines. Besides, the work has not tackled the issue of predicting the missing values in the skyline results. In other words, the skylines produced from the incomplete database are also incomplete and might be not useful and even misguide the user. We argue that providing the estimated value with reasonable relative errors against the real missing value is more useful to the user and helps him or her to make the right decision.

Bharuka and Kumar proposed another approach called Incomplete Data Frequent Skyline (IDFS) [27]. The *IDFS* approach aims to control the size of the skylines by relying on the concept of top-k and skyline techniques. The target of this approach is returning the superior skylines from datasets with missing values ordered by their fractional skyline frequency. Some experimental results of this work demonstrated the efficiency of *IDFS* when using both real and synthetic datasets. However, *IDFS* may not perform well and the performance highly degraded if the examined space for determining the frequent skylines tuples is very large.

Gao *et al.* [28] introduced two efficient algorithms for skyline query processing on incomplete data, in which tuples might have missing values in one or more attributes. These algorithms are *k-SkyBand* algorithms for incomplete data (*kISB*) and *Virtual Point-based* algorithm (VP), which include novel concepts such as the expired skyline, shadow skyline, and thickness warehouse that boost the search performance. The *kISB* algorithm is designed for *kSB* query (*k-Skyband query*) on incomplete data, which employs the concepts of thickness warehouse and expired skyline to improve the search performance. On the other hand, the *VP* algorithm utilizes the concept of virtual point, expired skyline, and shadow skylines to reduce the size of the candidate set and boost the query performance. These algorithms follow the same concept of [20]. Using the baseline algorithm, firstly the tuples are divided into different buckets as per bitmap representation and then some tuples called CkSB (candidate $k$slyband) are selected as the non-dominated tuples of each bucket. Finally, in the last stage of the baseline algorithm GkSB (global $k$skyband) the set of tuples is selected after cross-domination tests between CkBS tuples from different buckets. Due to the inefficiency of this algorithm, the research team developed a virtual *point-based* algorithm in which virtual points (VPs) are compared with cross-buckets instead of comparing all tuples of each CkSB. However, creating virtual points for each bucket may increase the redundancy of VPs and extra memory usage. Thus, another algorithm has been proposed (kISB) that helps prune unwanted CkSB as early as possible before executing the cross-domination tests.

The work in [21] introduced an algorithm named *Incosky-line* to process skyline queries for incomplete data attempting to avoid the issue of *cyclic dominance* in deriving skylines. Their proposed approach consists of the four phases of clustering data, grouping and identifying local skylines, generating k-dom skylines, and identifying incomplete skylines. Here, the initial database is divided into different clusters based on the bitmap representation of the tuples in which tuples with common non-missing attributes are grouped in one cluster. Without losing the *transitivity property*, tuples grouped in the same cluster can be easily compared. Then, the clusters are further grouped into small groups based on the highest value of any diminutions in the cluster before deriving local skylines. In the next phase, k-dom skylines derive a set of virtual skylines formed out of the local skylines of each cluster. The derived k-dom skylines are then combined to produce one global k-dom skyline followed by inserting the global k-dom skylines at the top of each cluster to prevent the dominated tuples from further processing. At last, in the retrieving skyline phase, the non-dominated local skyline points are processed by further comparing against each other to ensure that the final skyline encompasses the entire database. Their work focuses on reducing the number of pairwise comparisons and the searching space. This work also has not considered the issue of providing the skyline with complete data. It can thus be concluded that the work has only focused on processing skyline queries on incomplete traditional databases without taking into consideration the missing values present in the skylines. In addition, the work has been developed using traditional databases meaning that the approach does not fit crowd-sourcing databases which have different characteristics to traditional databases.

The work in [30] proposed the COBO framework which employs an ISSA algorithm that compute the skyline query process in two stages: *pruning compared list* and *reducing expected comparison times*. *Pruning compared list* uses the same approach as the *Bucket* algorithm [20] for pruning unwanted tuples from each partition. Before comparing crossed tuples, the stage of *reducing expected comparison times* finds the total sum of complete attributes of all non-dominated tuples from each bucket and sorts those tuples according in an ascending order (smaller value considered as best). This technique minimizes the number of comparisons and processing time.

The novel *skyline preference query* strategy is based on a massive and incomplete data set proposed by [29]. This strategy is named SPQ algorithm that divides massive and incomplete data set into two parts according to attribute importance and executes a skyline query. Secondly, a skyline preference query strategy based on loose clustering is implemented on attributes that have lower importance. Finally, the local skyline query results need to integrate. The experimental results based on synthetic data sets manifest that the SPQ algorithm is more efficient and achieves more positive result accuracy than other algorithms.

An approach for processing skyline queries in probabilistic incomplete databases have been proposed in [38] named EP. In probabilistic databases, some of attribute values are present in the range. EP algorithm has been designed based on the multi-level grouping idea. Where it started by adopts BUCKET algorithm [20] in order to divide the dataset into different buckets based on bitmap representation. Therefore, can sort the tuples in each buckets in descending order in a parallel way. Moreover, this approach applied pruning strategies, and pruning tuple transferring, that leads to reduce the computational costs. Lastly, final skylines are retrieving by comparing the local skylines of each bucket. However, this algorithm demonstrated to be efficient and valuable in this scope, where it decreased the processing time to a few seconds. Besides, it greatly decreases the time-cost by processing the high-dimensional large databases in a parallel way.

The work in [22] proposed a new algorithm called *Sorting-based Cluster Skyline Algorithm* (SCSA) that processes the skyline queries in incomplete cloud databases. In this algorithm, the initial database is divided into a set of clusters. Subsequently, the constructed clusters are further divided into smaller manageable groups to facilitate the skyline process and identify the local skylines of each cluster. The algorithm also employs the concept of generating *domination power* (*dp*) that eliminates many dominated tuples before retrieving the final skylines. Removing those dominated tuples before applying the skyline technique leads to saving a large amount of unnecessary pairwise comparisons and reduces the overhead of the skyline process. Moreover, to make this algorithm more efficient the algorithm includes an optimization process to eliminating some of the local cluster skylines before comparing them with the local skylines of other clusters. The experimental results demonstrated this algorithm is great superior and outperforming the current existing algorithms.

Three efficient methods for probabilistic skyline computation on incomplete data have been proposed in [39] called Sorting-based Probabilistic Skyline algorithm on incomplete data complying with Independent distribution (SPISkyline), Sorting-based Probabilistic Skyline algorithm on incomplete data complying with Correlated distribution (SPCSkyline) and Sorting-based Probabilistic Skyline algorithm on incomplete data complying with Anti-correlated distribution (SPASkyline). Both of SPISkyline and SPCSkyline algorithms employed pruning strategy, optimization of the process of probability computation, and sorting technique to improve the efficiency of probabilistic skyline computation. While the SPASkyline applying optimization of the process of probability computation, and sorting technique to improve the efficiency of probabilistic skyline computation. That because the pruning technique is ineffective for the incomplete data over anti-correlated database. Lastly, the experimental results show that the proposed algorithms are an effective method to process the skyline queries on incomplete data.

From the work reviewed in this section, the following has been concluded. Most of the previous works have only focused on how to tackle the issue of processing skyline queries in incomplete data aiming at solving the issue of *cyclic dominance* and losing the *transitivity property* of the skyline technique. However, not much attention has been paid to improving the quality of the skyline results by providing estimated values for the skylines with incomplete data. In fact, the only work that addresses the issue of predicting the missing values of the skylines is that contributed by [31] aiming at manipulating the missing values before returning the skylines to the user.

Table 1 summarizes the skyline techniques presented in this section.

B. SKYLINE QUERIES ON CROWD-SOURCED-ENABLED INCOMPLETE DATABASES

In recent years, query processing in crowd-sourcing databases has been explored and various techniques have been suggested to extend the traditional query operators such as select, join, group by, and maximum to support queries in the crowd [8], [9], [13], [40]. Due to the importance of skyline queries in the crowd-sourcing system, many research works have been proposed that aim tackle the issue of processing skyline queries in crowd-sourcing databases. To the best of our knowledge, all these previous studies address the issue of processing skyline queries in crowd-sourcing databases with incomplete data. This is due to the fact that processing skyline query with incomplete data can benefit from the crowd through estimating the missing values in the skylines. Most of the previous approaches concentrating on processing skyline queries in crowd-sourcing incomplete databases take into account reducing the monetary cost and the time latency while maintaining a high quality of skyline results.

TABLE 1. Summary of previous approaches of skyline techniques in incomplete database.

Author and Year	Approach	Data Distribution	Database Type	No. of Attributes	Missing Rate
Khalefa et al., [20]	Iskyline	Anti-cor., Cor., and Ind.	Synthetic, real	100 100 - 500	20% 10%-90%
Arefin & Morimoto [25]	RBSSQ	Ind., and Uni.	Synthetic	2 – 10	10%-40%
Bharuka & Kumar [26]	SIDS	Ind., and Anti-cor.	Synthetic, real	2 – 15 17	≥ 20% 10%-90%
Bharuka & Kumar [27]	IDFS	Anti-cor., Cor., and Ind.	Synthetic, real	10-25	20%
Gao et al. [28]	VP, kISB	Ind., and Cor.	Synthetic, real	20 - 100 17	5%-80% ≥ 20%
Alwan.et al. [21]	IncoSkyline	Anti-cor., Cor., and Ind.	Synthetic, real	6 – 12	10%-50%
Zhang et al. [30]	ISSA	Ind., and Cor.	Synthetic, real	6 - 6k	25%-50%
Wang et al. [29]	SPQ	Ind.	Synthetic, real	2 - 16 5	10%-50% ≥ 20%
Gulzar et al. [22]	SCSA	Anti-cor., Cor., and Ind.	Synthetic, real	5 - 13 4 - 20	4 – 12 attributes 3 – 19 attributes
Zeng et al. [38]	EP	Uni.	Synthetic, real	2 - 7	20%-60%
Zhang et al. [39]	SPISkyline, SPCSkyline, SPASkyline	Anti-cor., Cor., and Ind.	Synthetic, real	6 - 10	10%-90%

This section presents and examines the previous works relevant to processing skyline queries in crowd-sourcing incomplete databases.

The first work that addressed the issues associated with skyline queries in crowd-sourcing incomplete databases is contributed by the work in [5]. They proposed a hybrid technique that incorporates dynamic crowd-sourcing as part of a heuristic model. The idea of the proposed technique relies on utilizing a set of heuristic rules and employing *K-Nearest Neighbour* (*KNN*) to help in predicting the missing values for all tuples before applying the skyline process. The proposed estimation technique attempts to impute the missing values utilizing the available data in the database. Subsequently, it filters out the tuples by considering only those tuples with estimated values and high potential to be in the skylines for further processing. Besides, the proposed solution also attempts to exploit the contents of the crowd in order to enhance the accuracy of the estimated values. Thus, those tuples with predicted values generated from the crowd-sourcing database are compared with those generated using the local estimation. If tuples with the value estimated by the crowd have higher comparison accuracy with those values estimated using the local data, those estimated values generated from the crowd are preferably used in these tuples. However, the *K-NN* technique is not suitable when handling databases with a high missing data rate and a wide range of values. In this case, the quality of the predicted data highly influences the missing rate; if the database has a high missing rate, then the quality of the result will decline significantly [3], [34]. Moreover, the proposed solution attempt to generate estimated values for all missing data in the database before applying the skyline process. Doing so results in unnecessary monetary cost and leads to longer time latency due to the exhaustive process of estimating the missing data.

The work in [6] has been further extended in 2015 [7] focusing on the issue of skyline queries in incomplete web databases. The proposed approach incorporates some optimization techniques to compare the estimated values produced using the local data (offline) and the values estimated through the crowd. The proposed approach for local estimating attempts to offer an alternative solution when the crowd-sourcing may not be a feasible solution. This might occur if the cost of crowd-sourcing estimation is higher than the cost of local estimation or the quality of the local estimation is better than the quality of crowd-sourcing estimation. The local estimation approach relies on utilizing some advanced heuristics rules that focus on the correctness of the skyline results in order to reduce the amount of data to be estimated from the crowd. Besides, the crowd-based approach attempts to utilize the crowd-enabled databases to complete the process of estimating missing values. Crowd-based solution limits the estimation to only those tuples where heuristics (offline) will most likely introduce errors. Several experiments on real-life databases have been conducted to evaluate the effectiveness of the proposed approach taking into account the precision (P), recall (R), and skyline errors. In addition, the crowd-based approach has been evaluated to measure the relative errors taking into account the monetary cost and the time latency to generate the missing values. However, the correctness of the skyline might be deteriorated when relying on heuristics rules to identify the skylines. It may be that certain dominated tuples are included in the skyline results (false positive) and/or certain tuples which should be included in the skyline results are omitted (false negative) [4]. Furthermore, it is most likely that the relative error between the actual and the estimated values become very high if heuristics rules cannot capture the semantic relationship between the attributes; also if the missing rate in

TABLE 2. Summary of previous approaches of skyline techniques in incomplete crowdsourcing database.

Author and Year	Approach	Data Distribution	Database Type	No. of Attributes	Missing Rate
Lofi et al. [5]	Heuristic KNN	-	Real	6 - 8	1% - 20%
Lofi et al. [6]	Heuristic Min value	-	Real	6 - 8	1% - 20%
Lee et al. [32]	CrowdSky	Ind., and Anti-cor.	Synthetic, real	2 - 5	1 – 3 attributes
Miao et al.[41]	BayesCrowd	-	Synthetic, real	11 9	5% - 20%

the database is high as it impacts the quality of the estimated values. Most importantly, the proposed solution performs the value estimation process on the underlying data before applying the skyline process, which leads to unnecessary estimation for many unwanted data that do not contribute to the skyline results.

The work presented in [32] addressed the issue of processing skyline queries in the incomplete database with crowdsourcing. In their study, the proposed algorithm, *CrowdSky* relies on the crowd-sourcing database to estimate the missing values. The main purpose of the proposed solution is to exploit the power of the crowd in order to infer plausible values for attributes with missing values. The proposed approach emphasizes on the three different factors of monetary cost, time latency, and result accuracy when using the crowd as a source of value estimation. The *CrowdSky* algorithm incorporates pruning methods for eliminating unnecessary questions and thus minimizes the monetary cost of computing a crowd-sourced skyline. Secondly, the *CrowdSky* method also helps reduce the time latency imposed to generate the skylines from the crowd by limiting the questions to be asked to the crowd to independent questions and reducing the number of rounds by asking multiple questions in one round. A set of questions is called 'independent' if (and only if) the answers of each question in the same set do not affect the other questions. Lastly, *CrowdSky* also helps improve the accuracy of the crowd-sourcing skyline by assigning multiple workers for each question and then determine the final answer using the concept of majority voting. The majority voting technique attempts to dynamically assign the questions to the workers according to the level of importance of the question. Therefore, this dynamic assignment for workers can improve the accuracy of the crowd-sourced skyline with no additional monetary cost. Many experiments on synthetic and real datasets have been carried out to validate the performance and the effectiveness of the proposed approach, taking into account the three key factors (monetary cost, time latency, and result accuracy).

To the best of our knowledge, the work in [41] is the most recent work highlights the issue of skyline queries on crowd-sourced enabled incomplete database. They proposed an efficient framework for solving the skyline problem over the partial complete database with crowdsourcing called *Bayescrowd*. This framework included two stages; the modelling stage responsible to generate the conditions for each object that probably become a query answer. Those conditions presented by c-table. The second stage is the crowd that use many strategies to select the task which sends to crowd taking into consideration the cost and latency that given by the user. the experiment results on both real and synthetic datasets show the Bayes Crowd outperform the *CrowdSky* in reducing the execution time, monetary cost, and latency. The experiment results on both real and synthetic datasets show the Bayes Crowd outperform the *CrowdSky* in reducing the execution time, monetary cost, and latency. Besides both of *BayesCrowd* and *CrowdSky* based only on the crowd to estimate the missing values.

However, the researchers assumed that all missing values will be estimated by utilizing the crowd-sourcing databases. Thus, in the extreme case, there might be a large number of independent questions that need to be answered from the crowd in order to provide estimated values for those tuples with missing values. Hence, this large number of independent questions imposes high monetary cost and high time latency to generate the missing values. Most importantly, if the missing rate in the database is high, then a large number of missing values need to be sent to the crowd for estimation purposes. Therefore, a higher missing rate will incur a high monetary cost and longer time latency during the value estimation process. Table 2 summarizes the previous skyline techniques in the crowd-sourcing database that are presented in this section.

III. DEFINITIONS AND NOTATIONS

In this section, several definitions and notations are provided related to the skylines queries in incomplete databases. These definitions and notations are important to be clarified as part of our proposed approach. Our approach has been developed in the context of an incomplete relational database, D. A relation of the database D is denoted by $R(d_1, d_2, \ldots, d_m)$ where R is the name of the relation with m-arity and $d = (d_1, d_2, \ldots, d_m)$ is the set of dimensions.

Definition 1 (Incomplete Database): Given a database $D(R_1, R_2, \ldots, R_n)$, where R_i is a relation denoted by $R_i(d_1, d_2, \ldots, d_m)$, D is said to be incomplete if (and only if) it contains at least a data item p_j with missing values in one or more dimensions d_k (attributes); otherwise, it is complete.

Definition 2 (Skyline): This technique retrieves the skyline data items S in such a way that any skyline data item is not dominated by any other data item in the database.

Definition 3 (Dominance on Complete Database): Given two data items p_i and $p_j \in D$ database with complete data d

dimensions, p_i dominates p_j (denoted by $p_i \succ p_j$) if (and only if) the following two conditions hold:

1. $\forall d_k \in d, p_i.d_k \geq p_j.d_k$ **and**
2. $\exists d_l \in d, p_i.d_l > p_j.d_l$

Definition 4 (Dominance on Incomplete Database): Given two data items p_i and $p_j \in D$ incomplete database with d dimensions, p_i dominates p_j (denoted by $p_i \succ p_j$) if (and only if) the following three conditions hold:

1. The values of d_k and $d_l \in d$ for p_i and p_j must be non-missing **and**
2. $\forall d_k \in d, p_i.d_k \geq p_j.d_k$ **and**
3. $\exists d_l \in d, p_i.d_l > p_j.d_l$

Definition 5 (Skyline Queries on Complete Database): Select a data item p_i from the set of D complete database if (and only if) p_i is as good as p_j (where $i \neq j$) in all dimensions (attributes) and superior to p_j in at least one dimension. We use *Sskyline* to denote the set of skyline data items, $Sskyline = \left(p_i \forall p_i, p_j \in D, p_i \succ p_j\right)$.

Definition 6 (Skyline Queries on Incomplete Database): Select a data item p_i from the set of D incomplete database if (and only if) p_i is as good as p_j (where $i \neq j$) in all common non-missing dimensions and strictly better than p_j in at least one common non-missing dimension. We use *IncoDskyline* to denote the set of skyline data items on an incomplete database, $IncoDskyline = \left(p_i \forall p_i, p_j \in D, p_i \succ p_j\right)$.

Definition 7 (Comparable): Let the data items a_i and $a_j \in R$, a_i and a_j are comparable (denoted by $a_i \approx a_j$) if (and only if) they have no missing values in at least one identical dimension; otherwise a_i is incomparable to a_j (denoted by $a_i \not\approx a_j$).

Definition 8 (Transitivity): Given tuples $p, r, t \in D$, if $p \prec r$ and $r \prec t$, then $p \prec t$.

Definition 9 (Cyclic Dominance): Given tuples $p, r, t \in D$, if $p \prec r$ and $r \prec t$, then $t \prec p$.

Definition 10 [Approximate Functional Dependency (AFD)]: Given a relation R, a subset X of its attributes, and a single attribute a of R, we say that there is an *Approximate Functional Dependency* (AFD) between X and a, denoted by $X \dashrightarrow a$, if the corresponding functional dependency $X \rightarrow a$ holds on all but a small fraction of the tuples of R. The set of attributes X is called a determining set of A denoted by *dtrSet* (a).

IV. THE PROPOSED APPROACH

This section describes the phases of the proposed approach for processing skyline queries in incomplete crowd-sourcing databases. It comprises of the seven phases of Filtering Data, Analyzing Attributes, Generating Approximate Functional Dependencies (AFDs), Calculating the Strength of Probability Correlations between Attributes, Predicting the Missing Values in the Skylines, Assessing the Accuracy of Estimated Values and Identifying the Final Complete Skyline that are further explained in the subsequent subsections. Figure 1 illustrates the phases of the proposed approach.

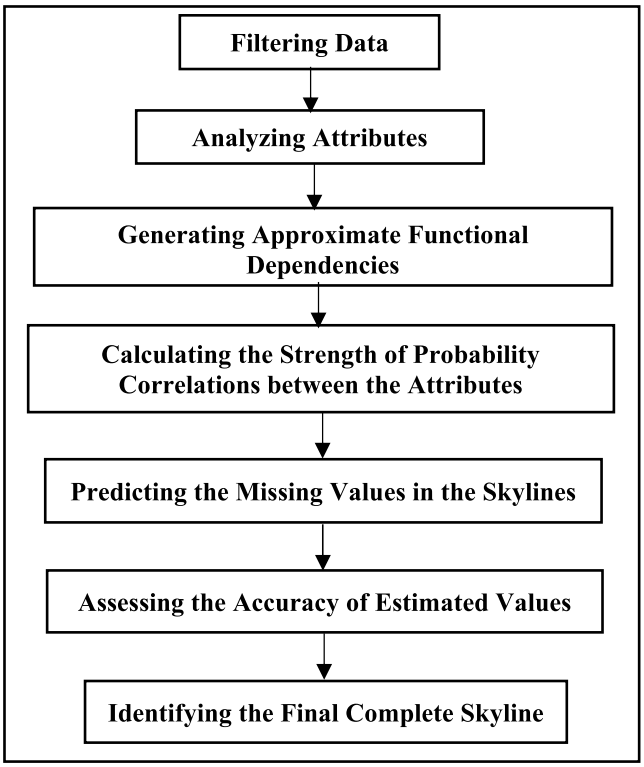

FIGURE 1. The phases of the proposed approach for processing skyline queries in incomplete crowd-sourcing databases.

To clarify the detail steps of the proposed approach and thoroughly explain the process of deriving skylines with complete data over incomplete crowd-sourcing databases, a running database example with incomplete data has been used as illustrated in Figure 2. The figure depicts that the database example encompasses 100 tuples with six (6) attributes (excluding the *PK* attribute). From the example, we notice that certain tuples in the database have missing values in one or more attributes. The missing values in the database have been denoted using the symbol '?' to indicate that the value of that particular attribute is not available. The *id* attribute represents the primary key of each tuple in the database and its value is always present. For instance, tuple m_6 (9, ?, 27, 19, 16, ?) denotes that the first, third, fourth, and fifth attributes have the values 9, 27, 19 and 16 respectively. However, the values for the second and sixth attributes are not present (missing).

It should be noted that the missing values of the database example have been randomly generated and can be found in one or more attributes for certain tuples. Besides, some tuples might have no missing values and the tuples are complete. For example, the following tuples have no missing values in any of the attributes (m_5, m_{10}, m_{19}, m_{24}, and m_{30}). In addition, the figure also depicts that the total number of cells in the entire database is 600 (excluding the primary key attribute). The number of cells with no missing values equal to 355 cells, while the number of cells with missing values is 245, representing 40% of the total amount of data. Thus, the database

Id	A_1	A_2	A_3	A_4	A_5	A_6
m_0	?	?	18	25	20	?
m_1	?	2	?	?	16	?
m_2	?	2	12	?	15	?
m_3	4	?	8	?	14	?
m_4	?	3	27	?	16	17
m_5	4	3	12	13	15	17
m_6	9	?	27	19	16	?
m_7	6	2	?	?	15	?
m_8	4	6	?	?	16	?
m9	8	2	?	?	16	?
m_{10}	6	4	25	25	18	19
m_{11}	?	?	18	25	20	?
m_{12}	?	2	?	?	15	17
m_{13}	8	2	?	?	16	?
m_{14}	?	3	27	?	16	17
m_{15}	?	?	12	?	?	?
m_{16}	9	2	27	19	16	18
m_{17}	4	2	?	9	?	?
m_{18}	?	?	18	25	20	?
m_{19}	8	2	16	17	16	18
m_{20}	?	2	12	?	?	?
m_{21}	?	3	27	?	16	17
m_{22}	?	3	?	?	?	?
m_{23}	?	?	18	25	20	?
m_{24}	4	2	8	9	14	16
m_{25}	?	2	?	?	16	?
m_{26}	?	2	?	?	?	18
m_{27}	?	2	12	?	15	?
m_{28}	8	?	?	17	16	?
m_{29}	?	2	?	?	16	?
m_{30}	6	4	25	25	18	19
m_{31}	?	2	12	?	15	?
m_{32}	6	2	12	?	?	?
m_{33}	?	?	18	25	20	?
m_{34}	4	3	12	13	15	17
m_{35}	6	4	25	25	18	19
m_{36}	?	2	16	?	?	?
m_{37}	6	?	?	?	?	19
m_{38}	?	?	12	13	15	?
m_{39}	4	6	?	?	?	?
m_{40}	?	3	27	?	16	17
m_{41}	?	2	?	?	15	17
m_{42}	8	2	?	?	16	?
m_{43}	8	?	16	?	?	?
m_{44}	8	?	24	25	19	?
m_{45}	?	?	16	17	?	18
m_{46}	?	3	27	?	16	17
m_{47}	4	2	8	9	14	16
m_{48}	4	?	?	?	15	17
m_{49}	?	?	?	25	?	17
m_{50}	?	2	16	?	16	?
m_{51}	?	?	18	25	20	?
m_{52}	8	2	16	17	16	18
m_{53}	6	?	25	25	18	?
m_{54}	?	2	?	17	?	?
m_{55}	4	2	8	9	14	16
m_{56}	?	3	27	?	16	17
m_{57}	6	?	25	25	18	?
m_{58}	6	2	12	25	15	17
m_{59}	5	2	?	11	?	?
m_{60}	?	?	18	25	20	?
m_{61}	?	3	?	?	?	17
m_{62}	4	2	?	9	?	?
m_{63}	9	?	27	19	16	?
m_{64}	5	?	?	11	12	?
m_{65}	?	?	18	25	20	?
m_{66}	6	2	12	?	?	?
m_{67}	8	2	16	?	?	?
m_{68}	?	2	?	?	16	?
m_{69}	?	?	18	25	20	?
m_{70}	4	2	?	9	?	?
m_{71}	9	?	27	19	16	?
m_{72}	?	?	18	25	20	?
m_{73}	6	2	12	?	?	?
m_{74}	9	2	27	19	16	18
m_{75}	?	2	16	?	?	18
m_{76}	6	2	12	?	?	?
m_{77}	?	3	27	?	16	17
m_{78}	6	4	25	25	18	19
m_{79}	?	2	16	?	16	?
m_{80}	?	3	?	25	?	17
m_{81}	4	2	8	9	14	16
m_{82}	?	3	27	?	16	17
m_{83}	6	2	12	?	?	?
m_{84}	4	2	8	9	14	16
m_{85}	?	?	18	25	20	?
m_{86}	5	3	?	?	?	?
m_{87}	6	2	12	?	?	?
m_{88}	9	?	27	19	16	?
m_{89}	8	2	16	17	16	18
m_{90}	6	?	25	25	18	?
m_{91}	?	2	16	17	?	?
m_{92}	9	?	27	19	16	?
m_{93}	4	2	?	?	14	?
m_{94}	?	?	18	25	20	?
m_{95}	8	2	16	17	16	18
m_{96}	6	4	25	25	18	19
m_{97}	?	2	8	?	?	?
m_{98}	?	2	?	?	16	?
m_{99}	?	3	27	?	16	17

FIGURE 2. The database with incomplete values.

has a missing rate of up to 40%. Lastly, the total number of tuples with no missing values is 19 tuples, and the number of tuples that have missing values in one or more attributes is 81 tuples. This indicates that more than 81% of the total number of tuples in the database has missing values.

A. FILTERING DATA

It has been argued that the size of the skyline is highly influenced by the total numbers of tuples and the number of attributes in the database relation [20], [21], [23]. This phase tries to remove all the dominated tuples from the initial database in the early stage which has no contribution to form the final skylines results. To this end, a skyline algorithm is performed on the incomplete database that eliminates those dominated tuples [36]. The purpose of this process is to remove these dominated tuples with missing values from further processing. Thus, removing these tuples with missing values before estimation saves monetary cost and reduces the

Algorithm 1

INPUT: An incomplete database, D
OUTPUT: A set of skylines with incomplete data, *IncoSky*

```
1. BEGIN
2.   FOR each tuple, t_i in D DO
3.     FOR each tuple t_j in D, where j <> i DO
4.       BEGIN
5.           IF tuple t_i > t_j  THEN
6.               Delete t_j from D
7.           ELSE
8.               IF tuple t_j > t_i THEN
9.                   Delete t_i from D
10.          END
11.      Insert tuples of  D into IncoSky
12.  END
```

FIGURE 3. The skyline algorithm.

complexity of the value estimation process for the missing values in the skylines. The skyline algorithm applied to the initial database is used to identify the non-dominated tuples by comparing all non-missing values of the tuple against each other using a pairwise comparisons process.

Figure 3 demonstrates the detailed steps of the skyline algorithm that has been adopted in the proposed approach for processing skyline queries in incomplete crowd-sourcing databases. Given the initial database with incomplete data, each tuple in the initial database D is analyzed (step 2). For each tuple t_i of the initial database D (step 3), if the tuple t_i dominates t_j (step 5), then t_i is deleted from the initial database D (step 6). Else, if the tuple t_j dominates the tuple t_i (step 7), then t_i is deleted from the initial database D (step 8). This process continues until all tuples in the initial database are compared against each other. Finally, the remaining tuples of the initial database are returned as skylines of the database, *IncoSky* (step 11). This process ensures that the returned tuples are the skylines of the entire incomplete database and all the eliminated tuples can be safely removed from further processing.

Figure 4 demonstrates the results of data filtration applied on the initial incomplete database. This initial step is very beneficial and leads to the elimination of those dominated tuples from further processing. The prior data pruning attempts to simplify the process of estimating the missing values in the next phases and limit the process of value estimation to those tuples that can contribute to the skyline formation. Hence, from the figure, we can observe that data pruning results into a significant reduction in the number of tuples to be considered during the missing value estimation process. Only 37 out of 100 tuples have been involved in the subsequent phases that translate into a reduction of up to 63% from the initial incomplete database.

Furthermore, this reduction in the number of tuples results into the decreased number of values to be estimated from 245 to 72 values or 70% reduction in the number of values to be estimated. Lastly, according to the figure, 30 tuples remain with incomplete data and seven (7) tuples with no missing

Id	A_1	A_2	A_3	A_4	A_5	A_6
m_0	?	?	18	25	20	?
m_4	?	3	27	?	16	17
m_6	9	?	27	19	16	?
m_{10}	6	4	25	25	18	19
m_{11}	?	?	18	25	20	?
m_{14}	?	3	27	?	16	17
m_{16}	9	2	27	19	16	18
m_{18}	?	?	18	25	20	?
m_{21}	?	3	27	?	16	17
m_{23}	?	?	18	25	20	?
m_{30}	6	4	25	25	18	19
m_{33}	?	?	18	25	20	?
m_{35}	6	4	25	25	18	19
m_{40}	?	3	27	?	16	17
m_{44}	8	?	24	25	19	?
m_{46}	?	3	27	?	16	17
m_{51}	?	?	18	25	20	?
m_{53}	6	?	25	25	18	?
m_{56}	?	3	27	?	16	17
m_{57}	6	?	25	25	18	?
m_{60}	?	?	18	25	20	?
m_{63}	9	?	27	19	16	?
m_{65}	?	?	18	25	20	?
m_{69}	?	?	18	25	20	?
m_{71}	9	?	27	19	16	?
m_{72}	?	?	18	25	20	?
m_{74}	9	2	27	19	16	18
m_{77}	?	3	27	?	16	17
m_{78}	6	4	25	25	18	19
m_{82}	?	3	27	?	16	17
m_{85}	?	?	18	25	20	?
m_{88}	9	?	27	19	16	?
m_{90}	6	?	25	25	18	?
m_{92}	9	?	27	19	16	?
m_{94}	?	?	18	25	20	?
m_{96}	6	4	25	25	18	19
m_{99}	?	3	27	?	16	17

FIGURE 4. Skylines of initial incomplete data (*IncoSky*).

values. Furthermore, the total number of missing values is 72 values and the total number of present values is 108 values. We argue that the data filtration process helps avoid many time-consuming and cost-prohibitive processes due to the fact that many unwanted tuples are successfully removed without compromising the correctness and the completeness of the skyline results.

TIME COMPLEXITY ANALYSIS OF ALGORITHM 1

Let $|D|$ denote the size of the database D, i.e., the number of data items in D, and let $|d|$ denote the dimensionality of a data item t_i. Let $|D| = n$ and $|d| = m$. Each basic operation $t_i > t_j$ or $t_j > t_i$ performs at least 1 and at most m comparisons. The outer loop (line 2) can repeat from 1 to $n-1$ times, and the inner loop (line 3) can repeat from 1 to $n-i$ times since $i <> j$. Then, the running time of the algorithm, denoted by $T(n, m)$, is between O (1) and O $\left(n^2 \times m\right)$, i.e.,

$$\mathrm{O}(1) \leq T_1(n, m) \leq \mathrm{O}\left(n^2 \times m\right).$$

Consequently, $1 \leq IncoSky \leq n$.

B. ANALYZING ATTRIBUTES

This phase aims to examine and analyze the available values in the initial incomplete database in order to identify the implicit relationships between the attributes. Identifying the implicit relationship between the attributes is considered as an essential step for generating the Approximate Functional Dependency (AFD). Capturing this implicit relationship between the attributes helps determine how the values in one attribute influence the values in the other attribute. This research work attempts to identify the relationships between the attributes using the Pearson linear correlation coefficient. The Pearson correlation coefficient is used to measure the strength of the relationship between two different variables and is denoted by r. The r value can take a range of values from 1 to -1. A value of 0 for r indicates that there is no relationship between the two variables, while a value greater than 0 indicates a positive relation. A positive relationship means that if the value of one variable increases, the value of the other variable also increases. In contrast, if the value of r is less than 0 the relationship is a negative relationship meaning that if the value in one variable increases, the value of other variable decreases [48]. The formula of the Pearson linear correlation coefficient applied to variable x and y is illustrated in Equation 1.

$$r_{xy} = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}} \quad (1)$$

where: n is the number of tuples, x and y are variable values of the attributes that measure the relationship between them.

In this regard, the formula given in Equation 1 is used to analyze the attributes of the database relation. Two different cases or scenarios might be produced that explains the relationship between the attributes. Exploring a relationship between different attributes indicates that the attributes are correlated to each other. Otherwise, no relationship is found and both attributes are independent of each other.

Figure 5 depicts the detailed steps of the algorithm for analyzing the attributes in our proposed approach. The algorithm input is the set of the skyline with incomplete data, while the output is the list of the relationships between the attributes of the database relation. We first analyze the values of the attributes A_i and A_j where $i <> j$ in order to determine the corresponding values of the correlation (step 2-3). Then, we compute the corresponding correlation between the attributes A_i and A_j using the Person linear correlation coefficient method (step 4 – 13). If the value of the correlation between A_i and A_j does not equal 0 (step 14), a new relationship is generated based on A_i and A_j and add it with its corresponding correlation strength $r\left(A_i, A_j\right)$ to *CorrAtt* (step 15). The process continues until all attributes are analyzed and compared against each other to identify the implicit relationships that exist between them. Finally, the set of relationships between the attributes is returned.

***Algorithm* 2**

INPUT: A set of skylines with incomplete data, $IncoSky$
OUTPUT: list of relationships between attributes with their strength value, $CorrAtt$

1. **BEGIN**
2. **FOR** each attribute A_i in *IncoSky* **DO**
3. **BEGIN**
4. $q = |IncoSky|$ // q is the number of tuples in $IncoSky$
5. Compute Sum_$A_i = \sum_{t_k \in IncoSky} t_k.v_i$ where v_i is the value of t_k for attribute A_i and it is available
6. Compute SSum_A_i = (Sum_A_i)^2
7. Compute Square_Sum_$A_i = \sum_{t_k \in IncoSky} t_k.v_i^2$ where v_i is the value of t_k for attribute A_i and it is available
8. **END**
9. **FOR** each attribute A_j *IncoSky* **DO,** where $j > i$ // since $r(A_i, A_j) = r(A_j, A_i)$
10. **BEGIN**
11. Compute Sum_$A_j = \sum_{t_k \in IncoSky} t_k.v_j$ where v_j is the value of t_k for attribute A_j and it is available
12. Compute Sum_A_i_A_j = Sum_A_i × Sum_A_j
13. Compute SSum_A_j = $\left(\text{Sum\_}A_j\right)^2$
14. Compute Square_Sum_$A_j = \sum_{t_k \in IncoSky} t_k.v_j^2$ where v_j is the value of t_k for attribute A_j and it is available
15. Compute $r(A_i, A_j) = \frac{(q \times \text{Sum\_}A_i\text{\_}A_j - \text{Sum\_}A_i \times \text{Sum\_}A_j)}{\sqrt{(q \times \text{Square\_Sum\_}A_i - \text{SSum\_}A_i) \times (q \times \text{Square\_Sum\_}A_j - \text{SSum\_}A_j)}}$
16. **IF** $(r(A_i, A_j) <> 0)$ **THEN**
17. Create a relationship based on A_i and A_j and insert it with its corresponding value of $r(A_i, A_j)$ to $CorrAtt$
18. **END**
19. **END**

FIGURE 5. Analyzing attribute algorithm.

TIME COMPLEXITY ANALYSIS OF ALGORITHM 2

Notice $1 \leq |IncoSky| = q \leq n = |D|$, and $|d| = m$ (i.e., the number of attributes). Since all the sums in lines (5-7) and lines (11-14) over all available attribute values of tuples t_i in *IncoSky*, the maximum number of terms in the sums is q. Then, we can estimate these all the operations in Algorithm 2 for each line involving additions and multiplications as follows:

Line 5: the number of additions = O (q)

Line 6: the number of multiplications = O (1)

Line 7: the number of additions = O (q) and the number of multiplications = O (q)

Similarly, we obtain the following estimations:

Line 11: the number of additions = O (q)

Line 12: the number of additions = O (q) and the number of multiplications = O (q)

Line 13: the number of multiplications = O (1)

Line 14: the number of additions = O (q) and the number of multiplications = O (q)

Line 15: the number of all operations = O (1)

Thus, in lines 5-7, the number of the operations is still O (q), and the number of operations in lines 11-15 is also O (q). Now, we can compute the total number of the operations, *Total* (m, q), used to construct all $r(A_i, A_j)$:

$$Total\ (q, m) = \sum_{i=1}^{m-1} \mathrm{O}\ (q) + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \mathrm{O}\ (q)$$

$$= \mathrm{O}\ (m \times q) + \mathrm{O}\left(m^2 \times q\right)$$
$$= \mathrm{O}\left(m^2 \times q\right).$$

Moreover, from lines 16-17, the number of comparisons and insertions, *ComIns* (m) can be measured as *ComIns* (m) = O (m^2). Then, the running time of Algorithm 2, $T_2\ (q, m)$ = O $(m^2 \times q)$ + O (m^2). Since $m \ll n$ and $q \leq n$, $T_2\ (n, m)$ = O $(m^2 \times n)$.

Based on Algorithm 2, the constructed correlation is considered positive if and only if the $r(A_i, A_j) > 0$. Conversely, the constructed correlation to be negative if and only if the $r(A_i, A_j) < 0$.

Based on our running database example, the result of analyzing attributes phase is described in Figure 6. The process of analyzing attributes starts by scanning the entire filtered incomplete database in order to discover the implicit correlation among the attributes. Then, we attempt to compute the correlation strength of the discovered relationships between the attributes using the formula given in Equation 1. Based on the analysis of our database example, we explore the following relationships $r\ (A_1, A_2)$, $r\ (A_1, A_3)$, $r\ (A_1, A_4)$, $r\ (A_1, A_5)$, $r\ (A_1, A_6)$, $r\ (A_2, A_3)$, $r\ (A_2, A_4)$, $r\ (A_2, A_5)$, $r\ (A_2, A_6)$, $r\ (A_3, A_4)$, $r\ (A_3, A_5)$, $r\ (A_3, A_6)$, $r\ (A_4, A_5)$, $r\ (A_4, A_6)$, $r\ (A_5, A_6)$. The figure also indicates the corresponding correlation strength of the discovered relationships that will be used in the next phase (Generating the Approximate Functional Dependencies (AFDs).

The Discovered Relationship	The Correlation Strength of the Discovered Relationship
$r(A_1, A_2)$	0.034889789
$r(A_1, A_3)$	0.518530738
$r(A_1, A_4)$	0.530290221
$r(A_1, A_5)$	0.039018517
$r(A_1, A_6)$	0.015630732
$r(A_2, A_3)$	0.565880443
$r(A_2, A_4)$	0.530047272
$r(A_2, A_5)$	0.498636516
$r(A_2, A_6)$	0.975446310
$r(A_3, A_4)$	0.606989858
$r(A_3, A_5)$	0.957319403
$r(A_3, A_6)$	0.609389558
$r(A_4, A_5)$	0.727933390
$r(A_4, A_6)$	0.591066008
$r(A_5, A_6)$	0.572369244

FIGURE 6. The list of discovered relationships between attributes with their corresponding strength.

C. GENERATING THE APPROXIMATE FUNCTIONAL DEPENDENCIES (AFDs)

In this phase, the AFDs are generated, which need to be derived based on the identified implicit relationship between the attributes [21], [42]. Referring only to the non-dominated tuples to generate the AFDs allows us to determine the relationship between the attributes. Typically, functional dependency is used to describe the explicit correlations between the attributes. Therefore, if some obvious and explicit correlations between the attributes are found, they can be directly added to the list of AFDs to be used for estimating the missing values in the non-dominated tuples. However, it is not always necessary that the explicit functional dependency can generate an accurate correlation between the attributes. For instance, in the car database example *Make* attribute is determined by *Model* attribute based on the given explicit functional dependency $Model \rightarrow Make$. However, given the $Model = 323$, the corresponding value for *Make* attribute should be *Mazda*, nevertheless, the value of *Make* attribute can also be *BMW* since the same value of 323 can also be found for models under *BMW* [7]. Therefore, it is not always necessary to use the explicit functional dependency as a main reliable source to estimate the missing values in the database. In crowd-sourcing databases, it is not always possible that the predefined functional dependencies that demonstrate the correlations between attributes are known. Thus, the idea of generating the AFD in this research work is the very interesting strategy that helps capture the implicit relationships between attributes aiming at estimating the missing values in the skylines.

It should be noted that the idea of our approach to generating the AFD relies on considering only the attribute correlations that have a strength value more than or equal to the user given threshold value. For simplicity and without lose generality, we assume that the user given threshold value should be not less than 0.50, ($T \geq 0.50$). The main reason behind this condition is due to the fact that the closer the value of the strength from 1 is the stronger the relationship between the attributes. By doing so, we guarantee that only correlations with high strength are considered for value estimation. This will ensure producing an estimated value with low relative error rate and sustain the quality of the value prediction. Thus, the generated approximate functional dependencies are based only on those strong correlations with a strength higher than the user given threshold value, T.

Algorithm 3

INPUT: List of relationships between attributes with their strength value, $CorrAtt$, user given threshold value for the acceptable attribute relationships, T.

OUTPUT: The set of AFDs, i.e., $AFD\_set = \{AFD_1, AFD_2, \ldots, AFD_n\}$

1. **BEGIN**
2. **FOR** each relationship $r(A_i, A_j)$ in $CorrAtt$ **DO**
3. **BEGIN**
4. $conf = r(A_i, A_j)$
5. **IF** $conf >= T$ **THEN**
6. Generate the corresponding AFD for A_i and A_j
7. Add the generated AFD into AFD_set
8. **END**
9. **END**

FIGURE 7. Generating the approximate functional dependencies.

Figure 7 illustrates the steps of generating the *Approximate Functional Dependency* (*AFD*) algorithm. The algorithm input consists of the list of generated relationships between the attributes together with their strength values and the user-given threshold value that determines the acceptable relationship between the attributes based on the corresponding correlation as computed in the previous step. Each generated relationship in the list of the relationships, *CorrAtt* has to be analyzed (step 2). We assume that *conf* denotes the value of the strength between the attributes A_i and A_j (step 4). This is followed by comparing the value of the strength of the corresponding relationship with the user-given threshold value. If the strength value of the relationship is greater than or equal to the user-given threshold value (step 5), the AFD of the corresponding relationship between the attribute A_i and A_j is generated (step 6). Furthermore, the generated AFD is subsequently added to the set of generated AFD, *AFD_set* (step 7). This process (steps 2 - 8) is repeated for every relationship in *CorrAtt* and the set of AFD, *AFD_set* is retrieved. By applying this process we can be certain that only attribute correlations with a high level of confidence are being considered when generating the corresponding AFDs between attributes which can be safely used to provide an accurate estimation for the missing values in the skylines due to their high level of correlation, while those attribute correlations with a value strength below the given threshold are being discarded from further processing, which in turn results into less time consumption and expenses.

TIME COMPLEXITY ANALYSIS OF ALGORITHM 3

The analysis of the algorithm is obvious: since $|CorrAtt| = m(m-1)/2$, checking each $conf = r(A_i, r_j)$ against the given threshold T and adding it to *AFD_set* takes $O(m^2)$ time,

i.e., the running time of Algorithm 3, $T_3(m)$, in the worst case, is $O(m^2)$ and in the best case, is $O(1)$. Consequently,

$$O(1) \leq |AFD\_set| \leq O\left(m^2\right).$$

Based on our running database example, the list of constructed attribute correlations given in Figure 6 will be further refined and only those correlations with a value strength higher than T are considered to generate the corresponding approximate functional dependencies. The new list of attribute correlations to be used for AFD are $r(A_1, A_3)$, $r(A_1, A_4)$, $r(A_2, A_3)$, $r(A_2, A_4)$, $r(A_2, A_6)$, $r(A_3, A_4)$, $r(A_3, A_5)$, $r(A_3, A_6)$, $r(A_4, A_5)$, $r(A_4, A_6)$, $r(A_5, A_6)$. Notice that the discovered relationship between A_1 and A_2 denoted as $r(A_1, A_2)$ has been ignored due to the low value of the correlation strength (0.034889789) in comparison with the user given threshold value ($T = 0.50$). Similarly, the discovered relationships $r(A_1, A_5)$ and $r(A_1, A_6)$ have also been disregarded due to the low value of the correlation strength in comparison with T. However, the relationships $r(A_1, A_3)$ and $r(A_1, A_4)$ have been considered for value estimation in the next phase. This is because the strength value of both correlations are higher than the user given threshold value, T. It should be noted here that the given relationship between A_1 and A_3 indicates that the value of A_3 determines the value of A_1. Similarly, the discovered correlation between A_2 and A_6 has been used for the value estimation in the next phase due to the high value of the correlation strength in comparison against the threshold value T (i.e., $0.97544631 \geq 0.50$). Therefore, the following AFDs remain to be used in the subsequent processes:

1. $\{A_3, A_4\} \dashrightarrow A_1$
2. $\{A_3, A_4, A_6\} \dashrightarrow A_2$
3. $\{A_4, A_5, A_6\} \dashrightarrow A_3$
4. $\{A_5, A_6\} \dashrightarrow A_4$
5. $\{A_5\} \dashrightarrow A_6$

D. CALCULATING THE STRENGTH OF PROBABILITY CORRELATIONS BETWEEN ATTRIBUTES

This phase attempts to calculate the strength of the relationship between the attributes in the database relation. The value of the strength reflects the degree of the correlation between the attributes, which in turn helps to identify an accurate estimation for the missing values in the skylines. Computing the strength of the probability correlation means utilizing the generated AFD between attributes in the previous step. This process helps to determine to what extent the value of the attribute A_i might influence the value of the attribute A_j.

In this paper, the strength of probability correlations between attributes A_i and A_j are represented as $P(A_i, A_j)$, the formula being as shown in Equation 2 [31], [33]:

$$P(A_i, A_j) = \frac{|A_j|}{|A_j \rightarrow A_i|} \quad (2)$$

Algorithm 4
INPUT: $AFD\_set = \{AFD_1, AFD_2, \ldots, AFD_n\}$
OUTPUT: The strength of probability correlations $P(A_i, A_j)$
1. BEGIN
2. FOR each $AFD_k(A_i, A_j)$ in AFD_set **DO**
3. $P(A_i, A_j) = |A_j| / |A_j \rightarrow A_i|$
4. END

FIGURE 8. Identifying the strength of probability correlations algorithm.

where $|.|$ represents the number of distinct values. The value of $P(A_i, A_j)$ denotes the strength that every distinct value in A_i is associated with a unique value in A_j.

From the previous step (generating AFD), we indicates the relationship between attributes A_i and A_j based on the non-missing values in both attributes by $AFD(A_i, A_j)$. We also assume that the attribute A_i is the attribute of interest which contains missing values. For every tuple p_l in the skyline set *IncoSky* the value of attribute A_i is not present, while the value of the attribute A_j is present. Therefore, the missing value of attribute A_i of tuple p_l can be estimated by exploiting the non-missing values of the attribute A_j for those tuples in *IncoSky*. This process generates several estimated values of the attribute A_i along with their relative frequencies. Subsequently, the generated frequencies are further analyzed to determine the value that has obtained the highest frequency. Thus, the value with the highest frequency is selected to be used as an estimated value for those missing values of attribute A_i. It might happen that there is more than one correlation that involves attribute A_i in the AFD_set. For example, one $AFD(A_i, A_k)$ between attribute A_i and A_k is listed in the AFD_set. If this is the case, then the process is repeated to generate the estimated values for the tuple p_l with their relative frequencies.

Figure 8 depicts the detailed steps of computing the strength of probability correlations between the attributes. The input of the algorithm includes the list of approximate functional dependencies produced in the previous step, AFD_set. The output of the algorithm comprises of the value of the strength of the probability correlation between the attribute A_i and A_j. The algorithm begins by analyzing each generated AFD that represents the correlation between attributes A_i and A_j in the AFD_set (step 2). This is followed by computing the value of the strength of probability correlations of attributes A_i and A_j. The computation is conducted based on the given formula in Equation 2 (step 3). This process continues until all AFDs in the AFD_set are analyzed and their strength values are computed.

TIME COMPLEXITY ANALYSIS OF ALGORITHM 4

Since $O(1) \leq |AFD\_set| \leq O(m^2)$, we can make the immediate conclusion that the running time of the algorithm, $T_4(m)$,

$$O(1) \leq T_4(m) \leq O\left(m^2\right).$$

To clarify the algorithm steps assumed in the *CAR* database, we assume a car with certain details (*mileage* = 40k, *power* = 600, *price* = null). Also, we assume that the *mileage* attribute has a correlation with the *price* attribute based on the generated AFD and its strength of probability correlation *P(mileage, price)* between those two attributes is 0.85. By analyzing the derived skylines with *mileage* = 40k we identify the estimated values for the *price* with their frequencies as (63k, 0.20), (60k, 0.25), (65k, 0.55). Hence, for the *price* attribute, we select the value 65k as the estimated value as it has the highest frequent value. However, in some cases, more than one AFD can be generated between the attributes. In this case, the results of the AFDs are combined to estimate the missing value. For instance, we assume that the car database contains a tuple with certain details (*mileage* = 60k, *power* = 700, *price* = null), where the *mileage* attribute has a correlation with the *price* attribute based on the generated AFD with strength is 0.75, and the *power* attribute and the *price* attribute have correlations based on the generated AFD with strength is 0.66. By analyzing the derived skylines with *mileage* = 60k, we obtain the following estimated values for the *price* with their frequencies (15k, 0.60), (14k, 0.40). Similarly, by analyzing the derived skylines for the *power* = 700 the *price* has the following values with their relative frequencies (15k, 0.45), (17k, 0.25), (14k, 0.30). Combining the obtained results for both attributes, to introduce an overall *price* with their strength probability ((15k, 0.747), (14k, 0.498), (17k, 0.165)), where, for example, the pair (15k, 0.747) is obtained by adding the results of the product of the frequencies of the pairs (15k, 0.60) and (15k, 0.45) with the probability strengths 0.75 and 0.66, respectively. That is, (0.60 * 0.75) + (0.45 * 0.66) = 0.747, whereby the value 15k is considered as the estimated value for the *price* attribute when the mileage is 60k since it has the highest frequency.

From the running database example, it can be learned that there is a relationship between A_1 and A_3 ($A_3 \rightarrow A_1$). This relationship indicates that the value of attribute A_3 can determine the value of the attribute A_1. Thus, based on this discovered functional dependency between A_1 and A_3, it is possible to determine the missing value of A_1 whenever the corresponding value of attribute A_3 is present. For example, notice that the tuple m_4 (?, 3, 27, ?, 16, 17) has a missing value for the attribute A_1 while the value of the attribute A_3 is present. Thus, we can estimate the missing value of A_1 in tuple m_4 by exploiting the available value of A_3. Here, the missing value of A_1 is estimated to be nine (9).

Nevertheless, if the corresponding value of A_3 is also missing, we should identify another relationship with attribute A_1. Thus, based on our database example, we observe that another relationship between A_1 and A_4 exists and we can employ the relationship $r(A_1, A_4)$ with the strength of probability correlation 0.530290221 to impute the missing value of A_1 for tuple m_{14}(?, 3, ?, 19, 16, 17). By analyzing the remaining tuples in the incomplete database, we conclude that when $A_4 = 19$, the corresponding value of A_1 can be either six (6) or eight (8). Then, based the frequency analysis for these both values we conclude that the value six (6) has a frequency of 0.87, while the value of eight (8) has a frequency of 0.13 with respect to $A_4 = 19$. We notice that the frequency of six (6) is higher than the frequency of eight (8) for the attribute A_1 when $A_4 = 19$. Thus, the value of six (6) is used as an estimated value for attribute A_1.

E. PREDICTING THE MISSING VALUES IN THE SKYLINES

This section explains the steps of predicting the missing values of the attributes in the skylines using the available data in the incomplete database. The idea of predicting the missing values relies on exploiting the AFDs that have been generated in the previous phases. Thus, the missing values of the attributes in the skylines can be easily estimated by benefiting from these AFDs which reflect the degree of the relationship between the attributes. The strengths of the probability correlation of the AFDs are also used to determine which AFD is most appropriate and can be used to provide an accurate estimation for the missing values of the attributes. During the estimation process, it may occur that multiple distinct values of a particular attribute are generated and need to be considered. In this case, we can select the most appropriate estimated value from the several alternative values by taking into account the frequency of the value. Hence, the value with the highest frequency is selected to replace the missing value of the attribute in the skyline. This case has been explained in the previous phase using a running example of a *CAR* database. We believe that the process of estimating the missing values using the local available data in the incomplete database is very beneficial as the local estimation with high accuracy is cheaper and leads to reasonable time latency in comparison with crowd-sourcing estimation. Accessing crowd-sourcing databases with the aim of estimating the missing values of the attribute in the skyline is expected to incur more cost and results in longer time latency to generate the estimated values. Most importantly, it is not always necessary that the quality of the estimated values using the crowd is acceptable and it is highly influenced by the quality of the crowd workers.

Figure 9 elaborates the steps of predicting the missing values algorithm that is being proposed in this research work. The input of the algorithm encompasses the set of the generated AFDs, *AFD_set*, along with their corresponding probability correlations strength, $P(A_i, A_j)$, and the set of skylines with missing values, *IncoSky*. The algorithm output consists of a list of estimated values of attributes with missing values in the skylines, denoted as *LocalEstVals*. The algorithm first identifies the set of all AFDs in *AFD_set*, denoted by $AFD(A_i)$, that can be used in finding the missing values in each attribute A_i in *IncoSky* (step 5). Next, it finds the set of all distinct values, $DistElems(A_i)$, of tuples in attribute A_i (step 6). The set of all distinct tuples of values in the attributes of ADFs corresponding to values in $DistElems(A_i)$ are defined in step 6. Further, for every value $v_{i,e}$ in each tuple $V = (v_{i,1}, v_{i,2}, \ldots, v_{i,k}) \in DistTuples(A_i)$, the algorithm computes the frequency $freq(v_{i,e}(u))$ of each

Algorithm 5

INPUT: The set of AFDs $AFD\_set = \{AFD_1, AFD_2, \ldots, AFD_n\}$, the strength of probability correlations of the AFDs, and the set of skylines with incomplete data, *IncoSky*.
OUTPUT: List of estimated values, $LocalEstVals$ to replace the missing values in *IncoSky*.

1. **BEGIN**
2. Initialize $ListEstVals = \emptyset$ // the list of estimated values to replace the missing values
3. **FOR** each A_i in $IncoSky$ if $AFD(A_i) \neq \emptyset$ **DO**
4. **BEGIN**
5. Let $AFD(A_i) = \{A_{i,1}, A_{i,2}, \ldots, A_{i,k} | A_{i,j} \dashrightarrow A_i \in AFD\_set,\ 1 \leq j \leq k\}$
6. Let $DistElems(A_i) = \{u_{i,1}, u_{i,2}, \ldots, u_{i,p}\}$ be the set of distinct elements in A_i
7. Let $DistTuples(A_i) = \{(v_{i,1}, v_{i,2}, \ldots, v_{i,k})\}$ be the set of distinct tuples in $(A_{i,1}, A_{i,2}, \ldots, A_{i,k})$ corresponding to some elements u's in $DistElems(A_i)$ // note, for some $1 \leq j \leq k$, $v_{i,j}$ may be empty if it is not used to identify the frequency of these u's
8. **FOR** each $V = (v_{i,1}, v_{i,2}, \ldots, v_{i,k}) \in DistTuples(A_i)$ **DO**
9. **BEGIN**
10. **FOR** each $v_{i,e} \in V$ **DO**
11. **BEGIN**
12. Compute the frequency $freq\left(v_{i,e}(u)\right)$ of each $u \in DistElements(A_i)$ corresponding to V
13. Compute the total frequency $totfreq_V(u)$ of each u: $totfreq_V(u) = \sum_{e=1}^{k} freq\left(v_{i,e}(u)\right) * P(A_i, A_{i,e})$
14. **END**
15. **END**
16. Select $u^* = \max_u totfreq_V(u)$ to be used for replacing the missing values of u
17. Add u^* to $LocalEstVals$
18. **END**
19. **END**

FIGURE 9. Missing values predicting algorithm.

$u \in DistElements(A_i)$ corresponding to V (step 12), then computes the total frequency of each u on V. Lastly, it selects the value u^* that correspond to the maximum total frequency as the replacement of missing value corresponding to V (step 16).

TIME COMPLEXITY ANALYSIS OF ALGORITHM 5

First, we notice that each attribute A_i in *IncoSky* may have missing values, and all the other attributes can be in ADF relation with A_i. Thus, $O(1) \leq |AFD(A_i)| \leq O(m)$. The number of missing values in each attribute A_i can be estimated in range $O(1) \leq |DistElems(A_i)| \leq O(|incoSky|) \leq O(n)$. The number of distinct tuples corresponding to distinct values in A_i can also be estimated as $O(1) \leq |DistTuples(A_i)| \leq O(n)$. Finding the distinct values in each A_i and corresponding distinct tuples in the rest attributes take $O(n)$ and $O(m \times n)$ times, respectively. For each value v in tuple $V \in DistTuples(A_i)$, corresponding to $u \in DistElems(A_i)$, the frequency is computed in $O(n)$ time. Consequently for V, the algorithm spends $O(m \times n)$ time, and for the total frequency, it takes $O(m)$ time. Thus, for all distinct values in A_i, Algorithm 5 takes $O(n) \times (O(m \times n) + O(m)) = O\left(n^2 \times m\right)$ time, since $m \ll n$. Lastly, for all attributes, i.e., for the algorithm, we obtain $O\left(n^2 \times m^2\right)$ as its time complexity, $T_5(n, m)$, i.e.,

$$T_5(n, m) = O\left(n^2 \times m^2\right).$$

Since for each attribute A_i, the number of missing values can be between $O(1)$ and $O(n)$. It then follows

$$O(1) \leq |LocalEstVals| \leq O(n \times m).$$

The final results of the value estimation using local data in the incomplete database for the missing values of the running example of the database using our proposed approach are given in Figure 10. However, these estimated values of the skylines are yet to be considered for replacement. First, the relative error between the real missing and the estimated values need to be calculated. If the relative error is less than or equal to the user-given threshold value for acceptable estimation, these estimated values are replaced with the missing value. Otherwise, the skylines will remain incomplete and further processing to provide an accurate estimation for those missing values by exploiting the crowd-sourced database has to be performed.

F. ASSESSING THE ACCURACY OF THE ESTIMATED VALUES

The next step of the proposed approach for processing skyline queries over crowd-sourcing incomplete databases is to assess the accuracy of the estimated values that have been generated in the previous steps. Assessing the accuracy of the estimated values means computing the relative error between the real missing and the estimated values of the attributes. A formal formula to compute the relative error between the actual missing value and the estimated value has been given in equation 4.

Id	A_1	A_2	A_3	A_4	A_5	A_6
m_0	6	4	18	25	20	19
m_4	9	3	27	19	16	17
m_6	9	3	27	19	16	17
m_{10}	6	4	25	25	18	19
m_{11}	6	4	18	25	20	19
m_{14}	9	3	27	19	16	17
m_{16}	9	2	27	19	16	18
m_{18}	6	4	18	25	20	19
m_{21}	9	3	27	19	16	17
m_{23}	6	4	18	25	20	19
m_{30}	6	4	25	25	18	19
m_{33}	6	4	18	25	20	19
m_{35}	6	4	25	25	18	19
m_{40}	9	3	27	19	16	17
m_{44}	8	4	24	25	19	19
m_{46}	9	3	27	19	16	17
m_{51}	6	4	18	25	20	19
m_{53}	6	4	25	25	18	19
m_{56}	9	3	27	19	16	17
m_{57}	6	4	25	25	18	19
m_{60}	6	4	18	25	20	19
m_{63}	9	3	27	19	16	17
m_{65}	6	4	18	25	20	19
m_{69}	6	4	18	25	20	19
m_{71}	9	3	27	19	16	17
m_{72}	6	4	18	25	20	19
m_{74}	9	2	27	19	16	18
m_{77}	9	3	27	19	16	17
m_{78}	6	4	25	25	18	19
m_{82}	9	3	27	19	16	17
m_{85}	6	4	18	25	20	19
m_{88}	9	3	27	19	16	17
m_{90}	6	4	25	25	18	19
m_{92}	9	3	27	19	16	17
m_{94}	6	4	18	25	20	19
m_{96}	6	4	25	25	18	19
m_{99}	9	3	27	19	16	17

FIGURE 10. Database with local estimation for the missing values.

The main aim of this phase is to ensure that only estimated values with high accuracy can be used to replace the missing values of the skylines which helps preserve the correctness of the skyline results and generates more insightful results that best fit with the user preferences. To determine whether the estimated values are accepted or not, a user-given threshold value is used which represents the minimum acceptable relative error for the estimated value. In this research work, we assume that the user-given threshold value falls within the range of 10% - 60%. Therefore, those estimated values with relative error less than or equal to the user-given threshold value can be safely used to replace the missing values. This process may result in replacing a large number of many missing values of the skylines by exploiting the local data in the database. Hence, an expensive and time-consuming process can be avoided due to the high monetary cost and the time latency required if these missing values are estimated using the crowd-sourcing databases. We believe that exploiting the available data in the database with the aim of estimating the missing values of the skylines is very beneficial as many missing values can be safely estimated instead of referring to the crowd. Nevertheless, in certain cases the relative error of the local estimated value of the skylines might be larger than the user-given threshold value. If this is the case it is better to refer to the crowd to produce more accurate estimated values of the skylines. The main reason behind not accepting the local estimated value with a high relative error rate and use the crowd-sourcing database as a source of estimating the values is to preserve the correctness of the skyline results.

The process of estimating the missing values using crowd begins by sending a request to the crowd with a set of predefined conditions including the constraints for the monetary cost, the accuracy of the result, and the time latency. In response, the crowd-sourcing platform attempts to identify the most appropriate workers that can provide accurate estimated values of the skylines before the given estimated values can be collected and returned to the user. If the user notices that the relative error of the crowd-sourced estimated value is better than the relative error of the local estimated value, the crowd-sourcing estimated value will be accepted. Otherwise, the user might ignore the crowd-sourcing estimated values and refer to the local estimated values which save cost. We conclude that it is no longer necessary that users have to accept the crowd-sourced estimated values in order to replace the missing values of the skylines due to the limited options. This is due to the fact that an alternative local estimated value can be used instead.

Figure 11 describes the detailed steps of the algorithm for assessing the accuracy of the estimated values using both local data and crowd-sourcing data. This process is accomplished by computing the relative error between the real missing value and the estimated value in order to determine whether the local estimated value or the crowd-sourcing estimated value can be used to replace the missing values of the skylines. The algorithm input includes the list of local estimated values, the actual missing values and the user-given threshold value of the relative error, while the output of the algorithm compares the skyline set with complete data. We first analyze each estimated value in the local estimation list, *LocEstVals* (step 2) and assume that the real missing value is known in advance and will be used to compute the relative error (step 4) before computing the relative error between the real missing value, $real_m$ and the estimated value e_m (step 5). If the relative error between values $real_m$ and e_m is less than or equal the threshold value of the relative error, $T_{relative\_error}$, then the estimated value e_m, is used to replace the missing value to the corresponding attribute in *IncoSky* (step 6-7). This process is repeated for all missing values in the skyline set, *IncoSky* which is expected to significantly reduce the number of values to be estimated from the crowd, which in turn result in a significant save in the monetary cost and time latency incurred due to the crowd-sourced estimation. However, in certain cases exploiting the local data in the database might not produce an accurate estimation for the missing value and may lead to unacceptable relative error between the real missing value and the estimated value. In such

Algorithm 6

INPUT: List of local estimated values of the missing values in $IncoSky$, $LocEstVals$, the actual missing values, a_v, The user given threshold value of the relative error, $T_{relative\_error}$.
OUTPUT: A list of skylines with complete data, $Complete\_Sky$

1. **BEGIN**
2. **FOR** each estimated value e_m in $LocEstVals$ **DO**
3. **BEGIN**
4. Let $real_m$ = the corresponding actual missing value of e_m
5. Compute the relative error between $real_m$ and e_m, $RE(a_m, e_m)$
6. **IF** $RE(a_m, e_m) <= T_{relative\_error}$ **THEN**
7. Replace the missing value of the corresponding attribute in $IncoSky$ with e_m
8. **END**
9. **FOR** each missing value in $IncoSky$ **DO**
10. **BEGIN**
11. Let $real_m$ = the corresponding actual missing value in $IncoSky$
12. Let $Crowd_e$ = the estimated value of the missing value in $IncoSky$ from the crowd
13. Compute the relative error between $real_m$ and $Crowd_e$, $RE(a_m, Crowd_e)$
14. **IF** $RE(a_m, Crowd_e) < RE(a_m, e_m)$ **THEN**
15. Replace the missing value of the corresponding attribute in $IncoSky$ with $Crowd_e$
16. **ELSE**
17. Replace the missing value of the corresponding attribute in $IncoSky$ with e_m
18. **END**
19. **END**

FIGURE 11. Assessing the accuracy of the predicted values for the missing values in the skyline algorithm.

cases, we have to refer to the crowd databases to obtain an accurate estimation for the missing values of the skyline set which cannot be derived using the local data. Steps 9 - 18 explain the process of estimating the remaining missing values in the skyline using the crowd-sourcing databases. It may occur that the relative error between the real missing value and the crowd-sourced estimated value is worse than the relative error between the real missing value and the local estimated value. In such a case it is better to replace the remaining missing values of the corresponding attributes in the skyline set with the local estimated value, e_m (steps 14 - 17). Notice that preferring the local estimated value to be used for the remaining missing values in the skyline set over the crowd-sourced estimated values results in lower monetary cost and more accurate estimated values. This shows that the idea of our proposed approach which relies on exploiting the local data in the database and analyzing the implicit relationships between the attributes to estimate the missing values in the skyline result is very beneficial as it helps reduce the number of values that need to be estimated from the crowd.

TIME COMPLEXITY ANALYSIS OF ALGORITHM 6

From the analysis of Algorithm 5, we have $\mathrm{O}(1) \leq |ListEstVals| \leq \mathrm{O}(n \times m)$. Since for each operation in lines 4-7, the algorithm spends O (1) time, consequently, for the first part (lines 2-8), it take at most $\mathrm{O}(n \times m)$ time. Similarly, in the second part (lines 9-18), it again spends at most $\mathrm{O}(n \times m)$ time. Thus, the time efficiency of the algorithm, $T_6(n, m)$, is

$$\mathrm{O}(1) \leq T_6(n, m) \leq \mathrm{O}(n \times m).$$

Referring to the running database example given in this paper, the relative error between the real missing and the estimated values is calculated using the formula given in equation 3 [3], [47]:

$$re_i = \frac{1}{a_i} . |a_i - e_i| \tag{3}$$

where the a_i represents the actual missing value while e_i is the estimated value

This relative error indicates the error rate between the obtained estimated values when the database has some missing values in one or more attributes and the real values when the database is complete with no missing values. For the experimental purposes, the final relative error is computed as given in equation 4.

$$re = \frac{\sum_{j=1}^{m} re_i}{m} \tag{4}$$

where m is the number of estimated values.

Figure 12 describes the relative error for every estimated value produced using the local estimation for our database example. For simplicity and without losing generality, we assume that the acceptable relative error for the estimated values should not be greater than 15%. The figure reveals that the relative error for most missing values in the skylines is zero which means that the prediction of these missing values using our *AFD*-based approach provides very

Id	RE(a_m, e_m) A_1	RE(a_m, e_m)A_2	RE(a_m, e_m)A_3	RE(a_m, e_m)A_4	RE(a_m, e_m)A_5	RE(a_m, e_m)A_6
m_0	0	33.33333333	0	0	0	0
m_4	0	0	0	0	0	0
m_6	0	50	0	0	0	5.555556
m_{10}	0	0	0	0	0	0
m_{11}	0	0	0	0	0	0
m_{14}	0	0	0	0	0	0
m_{16}	0	0	0	0	0	0
m_{18}	0	33.33333333	0	0	0	0
m_{21}	0	0	0	0	0	0
m_{23}	0	33.33333333	0	0	0	0
m_{30}	0	0	0	0	0	0
m_{33}	0	33.33333333	0	0	0	0
m_{35}	0	0	0	0	0	0
m_{40}	0	0	0	0	0	0
m_{44}	0	0	0	0	0	11.76471
m_{46}	0	0	0	0	0	0
m_{51}	0	33.33333333	0	0	0	0
m_{53}	0	0	0	0	0	0
m_{56}	0	0	0	0	0	0
m_{57}	0	0	0	0	0	0
m_{60}	0	33.33333333	0	0	0	0
m_{63}	0	50	0	0	0	5.555556
m_{65}	0	33.33333333	0	0	0	0
m_{69}	0	33.33333333	0	0	0	0
m_{71}	0	50	0	0	0	5.555556
m_{72}	0	33.33333333	0	0	0	0
m_{74}	0	0	0	0	0	0
m_{77}	0	0	0	0	0	0
m_{78}	0	0	0	0	0	0
m_{82}	0	0	0	0	0	0
m_{85}	0	33.33333333	0	0	0	0
m_{88}	0	50	0	0	0	5.555556
m_{90}	0	0	0	0	0	0
m_{92}	0	50	0	0	0	5.555556
m_{94}	0	33.33333333	0	0	0	0
m_{96}	0	0	0	0	0	0
m_{99}	0	0	0	0	0	0

FIGURE 12. The results of the relative error between the real missing and the estimated values of the skylines for the database example.

accurate estimation and we notice that the result accuracy is extremely high. Notice that the total number of estimated values of the skylines is 72. There are 50 estimated values produced with 0 relative error, while another six (6) estimated values are produced with a relative error of less than 15%. Hence, the total number of estimated values using local estimation which can be safely accepted totals 56, representing approximately 77% of the total number of values to be estimated. However, there are another 16 estimated values with a relative error greater than 15% which will not be accepted and should be estimated using the crowd-sourced databases. Notice that our proposed approach is scalable and able to provide a precise estimation for up to 77% of the entire amount of data to be estimated in the database example. Hence, we only need to estimate the remaining 16 missing values with high relative error using the crowd-sourced databases instead of 72 missing values.

This reduction in a number of values to be estimated using the crowd resources significantly decreases the monetary cost involved to estimate these missing values. Nevertheless, this does not mean that only the crowd can provide an accurate estimation for those submitted missing values. It may occur that the estimation using the local incomplete database produces a more accurate estimation than the crowd. In this case, we can reject the estimated values using the crowd and refer to those estimated values produced using our AFD-based approach. Hence, this will lead to a further reduction in the monetary cost and the time latency without compromising the accuracy of the estimated values. The final results of the relative error between the real and the missing values of the skylines is given in Figure 12. Figure 13 illustrates the list of tuples with missing values that have been estimated using the crowd-sourced databases. These estimated values have been accepted and replace the missing values of the skylines due to the fact that their relative error is better than the relative error of the local estimation.

G. IDENTIFYING THE FINAL COMPLETE SKYLINES

This section explains the last phase in the proposed approach for processing skyline queries on incomplete crowd-sourced

Id	A_1	A_2	A_3	A_4	A_5	A_5
m_0	6	3	18	25	20	19
m_6	9	2	27	19	16	17
m_{18}	6	3	18	25	20	19
m_{23}	6	3	18	25	20	19
m_{33}	6	3	18	25	20	19
m_{51}	6	3	18	25	20	19
m_{60}	6	3	18	25	20	19
m_{63}	6	3	18	25	20	19
m_{65}	9	2	27	19	16	17
m_{69}	6	3	18	25	20	19
m_{71}	6	3	18	25	20	19
m_{72}	9	2	27	19	16	17
m_{85}	6	3	18	25	20	19
m_{88}	6	3	18	25	20	19
m_{92}	9	2	27	19	16	17
m_{90}	6	3	18	25	20	19

FIGURE 13. Crowd-sourced estimated values.

databases. In this phase, we attempt to determine the new final skylines after replacing the missing values with the estimated values. This is due to the fact that it is not necessary the skyline results with missing values remain the same after replacing the missing values with the estimated values. Thus, this step is important to ensure that the retrieved skylines are correct and complete and no other tuple in the entire database can dominate these skylines. Hence, we have to apply any skyline algorithm that best fit with the complete database aiming at generating the new skyline results with complete data. The result of this phase represents the final skylines that will be returned to the end user. We argue that skylines result with complete data is more insightful and can guide the user to make a wise decision when working on a database with incomplete data. In this phase, we try to identify the final skyline result after completing the initial incomplete dataset.

Figure 14 presents the final skylines with complete data for the running incomplete database example. We believe that providing skylines with no missing values is more beneficial and assist the user in making the most appropriate decision that best fit his or her preferences. It should be noted that the correctness and the completeness of the skylines with the estimated values produced by our proposed solution have been ascertained. This is explained as follows. Since our proposed solution adopted the skyline technique proposed in [22] to filter out the initial incomplete database. Therefore, the correctness and the completeness of the initial skylines with incomplete data derived in our work have been validated and proved by the work in [22] and it is no longer necessary to be validated again. Most importantly, the correctness and the completeness of the skylines with the estimated values are validated when comparing the skylines with the estimated values against each other to derive the final skylines of the database as explained in the last phase of the proposed approach "*identifying the final complete skylines*". Thus, we guarantee that the reported skylines are not dominated by any other data items and they are the skylines of the entire incomplete database.

d	A_1	A_2	A_3	A_4	A_5	A_6
m_0	6	3	18	25	20	19
m_4	9	3	27	19	16	17
m_{10}	6	4	25	25	18	19
m_{11}	6	3	18	25	20	19
m_{14}	9	3	27	19	16	17
m_{16}	9	2	27	19	16	18
m_{18}	6	3	18	25	20	19
m_{21}	9	3	27	19	16	17
m_{23}	6	3	18	25	20	19
m_{30}	6	4	25	25	18	19
m_{33}	6	3	18	25	20	19
m_{35}	6	4	25	25	18	19
m_{40}	9	3	27	19	16	17
m_{44}	8	4	24	25	19	19
m_{46}	9	3	27	19	16	17
m_{51}	6	3	18	25	20	19
m_{53}	6	4	25	25	18	19
m_{56}	9	3	27	19	16	17
m_{57}	6	4	25	25	18	19
m_{60}	6	3	18	25	20	19
m_{65}	6	3	18	25	20	19
m_{69}	6	3	18	25	20	19
m_{72}	6	3	18	25	20	19
m_{74}	9	2	27	19	16	18
m_{77}	9	3	27	19	16	17
m_{78}	6	4	25	25	18	19
m_{82}	9	3	27	19	16	17
m_{85}	6	3	18	25	20	19
m_{90}	6	4	25	25	18	19
m_{94}	6	3	18	25	20	19
m_{96}	6	4	25	25	18	19
m_{99}	9	3	27	19	16	17

FIGURE 14. The final skyline result.

V. EXPERIMENT EVALUATION

This section explains the parameter settings of the experiments that have been carried out in this research work in order to evaluate the performance of the proposed approach presented in this thesis. In order to fairly evaluate the performance of the proposed solution and estimate the missing values of the skylines in crowd-sourcing enabled, incomplete databases, we compared our technique with the most recent technique for value estimation of skylines designed for crowd-sourced enabled incomplete database as introduced in [5]. To the best of our knowledge, their study constitutes.the only available study that addresses the issue of value estimation in such databases. Their approach of value estimation of skylines exploits the most prominent machine learning-based technique called K-Nearest Neighbor (*KNN*) to generate the missing values using the available data in the incomplete database. All approaches were implemented using Java programming language employing the Eclipse platform. Comprehensive experiments have been carried out on a computer with the following specifications: CPU Intel(R) Core(TM)-4510U i7 CPU @ 2.00 GHz 2.60 GHz with 8GB memory and Windows 10 Pro. As reported in the database literature, estimating the values on a database with a large size of incomplete data constitutes an expensive and time-consuming process [5], [7], [22], [31], [33], [35], [43].

TABLE 3. The parameters setting of the synthetic and real datasets.

Parameter Settings				
Dataset	Total No. of Attributes	Total No. of Attributes with Missing Values	Missing Rate (%)	Dataset Size (K)
Independent	7	1 – 4	10% - 60%	20K, 40K, 60K,80K, 100K
Correlated	7	1 – 4	10% - 60%	20K, 40K, 60K,80K, 100K
NBA	7	1 – 4	10% - 60%	4K, 8K, 12K,16K, 20K
Car	6	1 – 4	10% - 60%	4K, 8K, 12K,16K

Therefore, in our experiments, the performance metrics that have been considered are number of missing values of skylines to be estimated, the processing time for value estimation in the skyline, the relative error between the real missing value and the estimated value, outsourced value estimation, the cost of crowd-sourced value estimation, in addition to the time latency of crowd-sourced value estimation. Utilizing the crowd allows us to exploit the available data in the crowd databases and involves certain crowd workers to generate the estimated values of skylines. Thus, this process incurs the monetary cost and causes time latency. Thus, the main reason for using the monetary cost and time latency metrics in our research work to estimate the missing values of skylines is due to the fact that these metrics are the most critical factors when dealing with crowd-sourced databases [4], [5], [8], [10], [40], [44]–[47]. These six metrics are measured by varying the dataset size, missing rate, the user-given threshold value for the acceptable relative error rate between the real missing and the estimated values of the skylines, and the number of the batch for crowd-sourced estimation. In this thesis, two different types of the dataset have been used in the experiments, namely synthetic and real datasets. For the synthetic dataset, two different data distribution has been assumed and generated to run the experiments in the form of correlated and independent datasets. We have decided to choose the correlated and independent datasets due to the fact that these types represent the real world database and is most frequently used in value estimation for skylines on a database with incomplete data [26], [31], [32], [47]. Moreover, two types of real datasets have been used which are NBA and Car databases. These real datasets are initially complete with no missing values; nevertheless, we have manipulated these datasets to be incomplete by randomly removing certain attribute values [4], [6], [20], [23], [27], [36]. The parameter setting of both the synthetic (correlated and independent) and real (NBA and Car) datasets have been described in Table 3. For simplicity and without losing generality, we assume that the query condition in all experiments considered in this research work is to retrieve the skylines from the incomplete database with the maximum values.

A. EXPERIMENT RESULTS OF SKYLINES VALUE ESTIMATION

In this section, we present and discuss the experimental results of the synthetic and real datasets for the proposed approach for estimating skyline values on crowd-sourcing enabled incomplete databases. The main purpose here is to investigate and examine the impact of data filtration on the process of estimating accurate values for the missing values of the skylines on a database with incomplete data. Besides, we also try to examine the impact of the missing rate on the relative error between the real missing and estimated values. Moreover, the experiment is used to investigate the impact of the dataset size on the produced relative error between the real missing value and the imputed value. Furthermore, we also intend to study the effect of the user-given threshold value for the acceptable relative error rate between the real missing and the imputed values of the skylines. Last but not least, this section attempts to explore the relationship between the preferred user-given relative errors with respect to the monetary cost and the time latency of producing the estimated values from the crowd-sourced databases. Further details pertaining to these experiments are given in the following subsections.

1) EFFECT OF DATA FILTRATION

It is suggested that the process of generating precise value estimation on a large volume database with a high missing rate constitutes an exhaustive and time-consuming process [20], [23], [26], [29], [31], [33], [40]. This is due to the fact that the process of estimating the missing database values requires the scanning and examining of all the available data in order to capture the implicit relationships between the attributes that help produce a precise estimated value. Therefore, it is not desirable to perform the process of value estimation on the entire data contained in the database and rather perform data filtration first. This is particularly recommendable for skyline query processing on a database with incomplete data in which certain tuples with missing values may not be needed. Also, any estimation of these unwanted tuples incurs extra overhead for the process of value estimation of the skylines. Hence, this set of experiment highlights the impact of the data filtration on the number of missing values to be estimated and the processing time of the value estimation process. In this section, we depict the experimental results for both synthetic and real datasets that have been used in this research work. Figures 15(a), 15(b), 15(c), and 15(d) illustrate the results for the independent, correlated, NBA, and Car datasets respectively. For all experiments, the size of the dataset is varying, while the missing rate and the number of attributes are fixed.

The Figures 15(a) and 15(b) show the results for correlated and independent datasets in which the dataset size varies

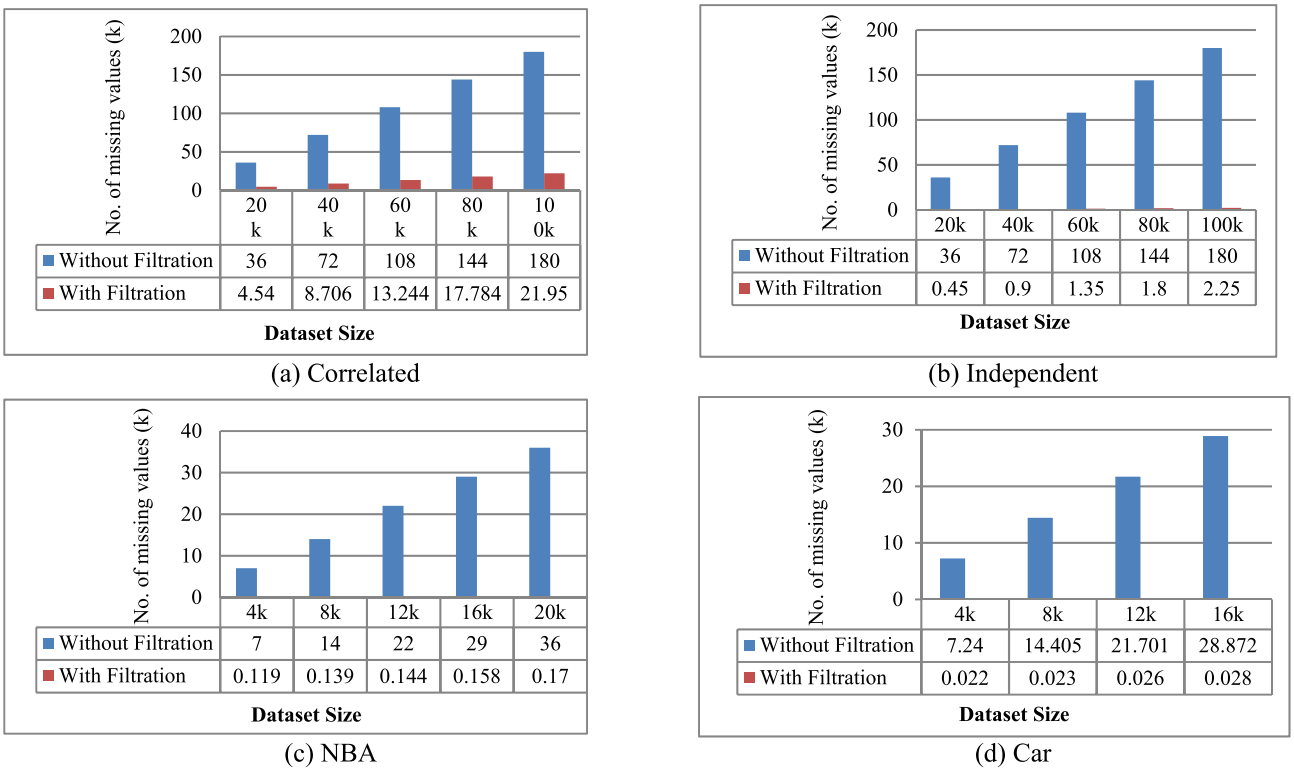

FIGURE 15. The effect of data filtration on the number of missing values to be estimated.

within the range of 20k-100k, while the missing rate is fixed to 30% and the total number of attributes fixed to six (6). The figures show that the concept of data filtration applied on both synthetic datasets reduces the rate of the number of missing values to be considered for estimation up to 89% of the total number of missing values in the initial incomplete dataset. Furthermore, Figure 15(c) demonstrates the experimental results for the NBA real dataset depicting the total number of missing values to be considered for the estimation process with and without filtration. For this set of experiment, the size of the dataset varies between 4k and 20k, while the missing rate and the number of attributes are fixed to 30% and six (6) attributes respectively. The results illustrate that data filtering indeed decreases the rate of the number of missing values for the NBA dataset up to 97%. Lastly, Figure 15(d) presents the experimental results on the Car dataset where the size of the dataset varies between 4k and 16k, and the missing rate is fixed to 30% and the number of attributes is fixed to four (4). The figure suggests that the proposed idea of applying the skyline technique before the value estimation process leads to a significant reduction in the number of values to be estimated.

From the results, it is obvious that the idea of data filtration that exploits the skyline technique on the initial incomplete database is very beneficial and results into a great reduction in the number of values to be considered for estimation. These missing values have been neglected due to the fact that they have no contribution to form the final skylines of the incomplete database. Thus, we can safely remove them before applying the process of value estimation for the missing skyline values. From the experimental results, we can also conclude that there is a strong relationship between the size of the database and the total number of missing values to be estimated. This means that the number of missing values gradually increases when the size of the database increases. Hence, we can conclude that the proposed idea of data filtration is extremely useful for the incomplete database with a large volume and can assist in simplifying the process of estimating the missing skyline values.

Figures 16(a), 16(b), 16(c), and 16(d) show the total processing time of the value estimation process with the proposed idea of data filtration and without data filtration for correlated, independent, NBA, and Car datasets respectively. The parameter settings of this set of experiments for all datasets are the same as the previous experiments reported in Figure 15. Figure 16(a) and 16(b) illustrate that data filtration results in a significant reduction in the processing time of the value estimation process for both synthetic datasets. We can notice that the processing time gradually increases when the size of the database increases. This is very obvious when the size of the database increases to 100K and the processing time of the value estimation for both synthetic datasets increases up to 175 minutes and 75 minutes respectively. However, our approach maintains a greatly reduced processing time for the value estimation process, and the size of the database has no significant impact on the process

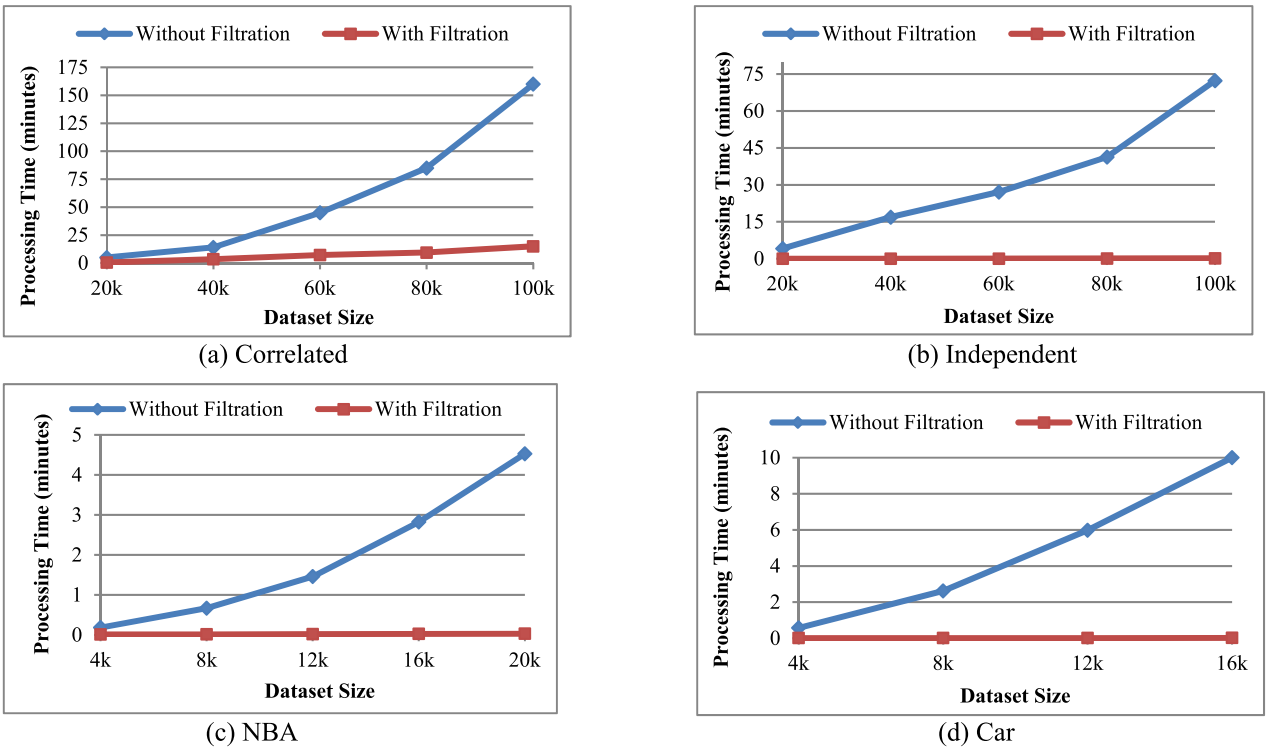

FIGURE 16. The effect of data filtration on the processing time of value estimation of skylines.

of value estimation. This is because the data filtration helps eliminate many unnecessary missing values from the value estimation process, which in turn results in minimizing the processing time. From the figure, we can also observe that the processing time of value estimation, without data filtration, dramatically increases when the size of the database increases. Moreover, Figures 16(c) and 16(d) indicate that our idea of data filtration assists in simplifying and speeding up the process of value estimation for both real datasets. We can thus conclude that our proposed approach for value estimation is superior in all cases, while the processing time of value estimation slightly increases when the database size increases. In contrast, we also find that the processing time of value estimation without data filtration dramatically increases when the database size increases. This is due to the fact that all the missing values in the initial incomplete database have been considered in the estimation process. Nevertheless, we argue that not all predicted values in the incomplete database are needed when the skylines are identified. Thus, most of these estimated values are unwanted, and tuples dominated by the skylines tuples prior to the value estimation process can be avoided to save the processing time. Also, Figures 16(c) and 16(d) indicate that the processing time of value estimation using our proposed data filtration process for NBA and Car real datasets remain below one (1) minute. However, the processing time of value estimation without data filtration for the NBA and Car datasets can take up to five (5) minutes and ten (10) minutes respectively.

2) IMPACT OF THE MISSING VALUE RATE

This set of experiments aims at studying the effect of the missing value rate on the relative error existing between the real missing value and the estimated values of skylines. Most of the literature in the area of handling missing data in the database reports that there is a significant impact of the missing value rate on the accuracy of the imputed values [1]–[3], [6], [31], [32]. Thus, this section attempts to examine the efficiency of the proposed approach when predicting the missing values of the skylines in terms of the missing value rate. Figures 17(a), 17(b), 17(c), and 17(d) demonstrate the relative error rate between the real missing value and the estimated value of skylines for the correlated, independent, NBA, and Car datasets, respectively. The experiments focus on examining the impact of the missing value rate on the relative error rate. The parameter settings of the synthetic dataset (correlated and independent) determine that the size of the dataset is fixed to 60k and the number of attributes is fixed to six (6), while the missing value rate varies between 10 and 60%. Furthermore, the real dataset (NBA and Car) size is fixed to 12k for both datasets, and the number of attributes is fixed to six (6) attributes for the NBA dataset and four (4) attributes for the Car dataset, and the missing value rate varies between 10 and 60%. The experimental results presented in the figure denote that the missing value rate has a significant impact on the relative error between the real missing value and the estimated value. This is clear for all cases in each experiment.

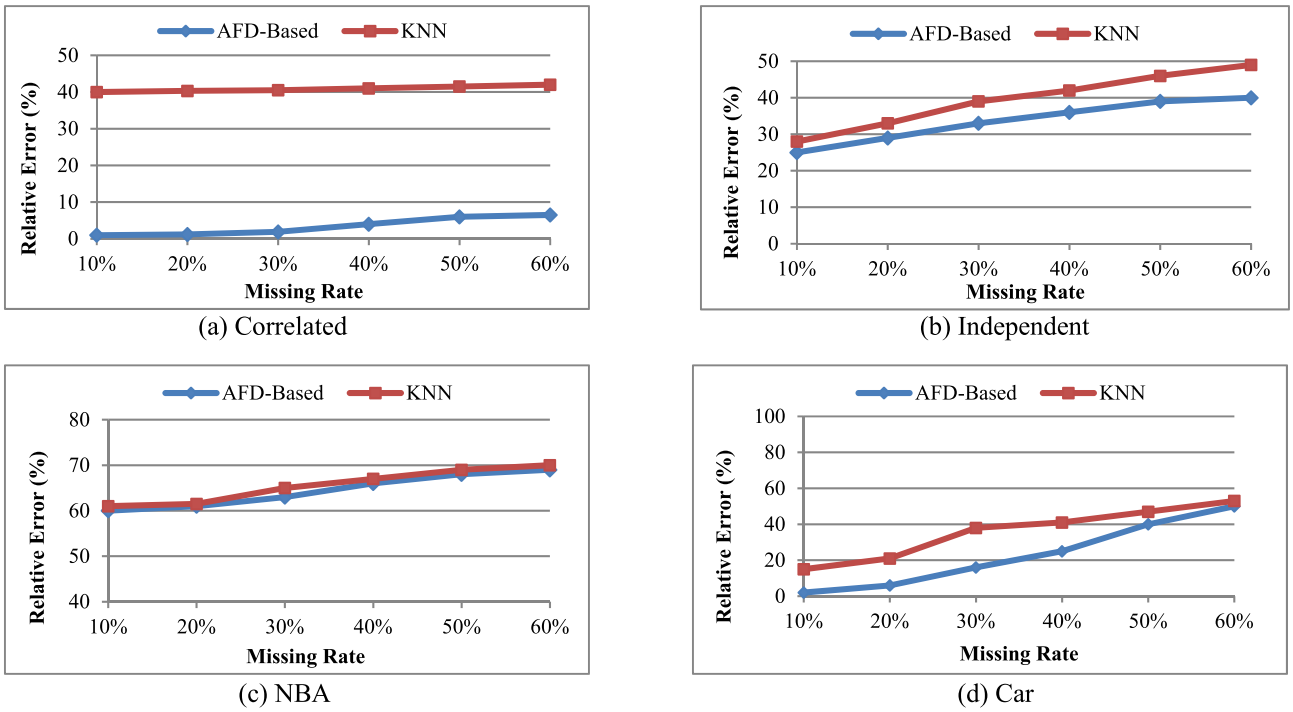

FIGURE 17. The effect of missing value rate on the relative error rate between the real missing value and the estimated value of the skylines.

Figure 17(a) demonstrates the experimental result of the relative error between the real missing and the estimated values on the correlated synthetic dataset. The result indicates that our proposed approach outperforms the previous approach by producing relative error less than 10% in all cases. Moreover, the figure also illustrates that the relative error is marginally increased when the missing rate is 40%. The main reason behind this significant improvement in the relative error rate in our approach is due to the fact that exploiting the available data to generate the AFD assists in producing an accurate estimation for the missing values in the database. In contrast, the result depicts that the relative error rate between the real missing value and the estimated value using the *KNN* technique is more than 40% in all cases. Figure 17(b) explains the experimental results of the relative error between the real missing and the estimated values of the skylines on the independent synthetic dataset. The results indicate that our *AFD*-based approach is superior and outperforms the *KNN* in all cases. Besides, the results also show that the relative error gradually increases when the missing rate increases. From the results, we can also conclude that the relative error between the estimated values and the real missing values is high in comparison to the results of the correlated synthetic dataset. This is due to the fact that for independent data it is very difficult to capture the implicit relationship between the attributes and the generated AFD, which may have low accuracy. Thus, it is very difficult to provide an accurate value estimation for the independent database with incomplete data. Nevertheless, our approach is performing better than *KNN* on the independent dataset for all cases.

Furthermore, Figure 17(c) presents the experimental results of the relative error that have been carried out on the NBA real dataset. We can notice that our approach outperforms the previous *KNN* approach in all cases by generating a lower relative error rate between the real missing and the estimated values. The results also show that the relative error produced by both approaches (*AFD*-based and *KNN*) falls within the range of 60% to 70% when the missing value rate is between 10 to 60%. The main cause of this high relative error between the real missing and the estimated values is due to the fact that the NBA real dataset is considered as a semi-independent dataset meaning that the missing rate for this set of experiment is high, which in turn leads to imprecisely estimated missing value. However, our approach manages to maintain a lower relative error in comparison to *KNN* in all cases.

Figure 17(d) describes the results of the experiment that have been generated on the Car real dataset. From the results, it can be seen that our proposed approach steadily outperforms the previous existing approach in all cases. The results also show that our approach produces a relative error of less than 10% when the missing rate is less than 20%. Nevertheless, the rate of relative error gradually increases when the missing rate is increasing. We also notice that the relative error produced by *KNN* is worse than our approach when the missing rate is less than 50%.

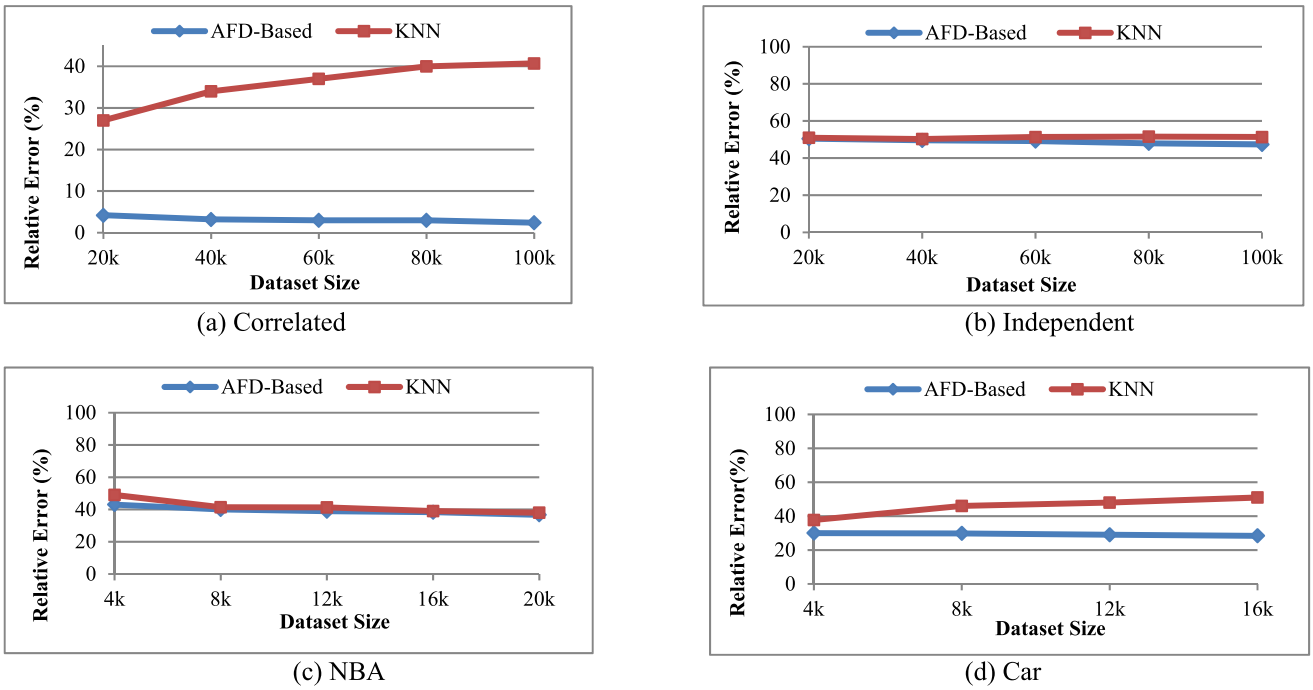

FIGURE 18. The effect of dataset size on the relative error rate between the real missing value and the estimated value of the skylines.

From the experiment results reported in this section we can conclude that there is a significant relationship between the missing rate of the database and the quality of the estimated values of the skylines. Most importantly, we can also observe that the data characteristics have a very significant impact of the relative error produced between the real missing values and the estimated values. In other words, it is very difficult to generate a precise estimation of the missing values contained in the independent and semi-independent databases. In contrast, it is much easier to obtain an accurate estimation for correlated and semi-correlated datasets using our proposed approach.

The results also demonstrate that the approach of exploiting the existing data in the database and analyzing the implicit relationships between attributes is very beneficial in providing a local and accurate estimation for the missing values. However, the results also illustrate that the *KNN* technique may not be appropriate for correlated and semi-correlated datasets with high missing rates. This is due to the fact that the distance between the neighboring values is longer when the missing rate increases, which in turn leads to a higher relative error rate between the real missing and the estimated values.

3) IMPACT OF DATASET SIZE

This section highlights the investigating the effect of the dataset size on the relative error between the real missing value and the estimated value of the skylines. It has been agreed in the database literature that the size of the database has a considerable impact on estimating the missing values in the database [20]–[22], [42]. Therefore, we try to examine the impact of dataset size on the produced relative error rate between the real missing value and the predicted value of the skylines by our *AFD*-based approach in this section.

Figures 18(a), 18(b), 18(c) and 18(d) illustrate the experimental results of the relative error rate for the correlated, independent, NBA, and Car datasets respectively. The experiments concentrate on examining the influence of the dataset size on the generated relative error rate between the real missing value and the imputed value of the skylines. For both synthetic datasets (correlated and independent) the dataset size varies between 20k and 100k, the number of attributes is fixed to six (6), and the missing value rate is set at 30% of the total size of the dataset. In addition, the parameter setting of the real dataset encompassing NBA and Car datasets determines the number of attributes for the NBA dataset as being six (6), while the number of attributes for the Car dataset as being four (4). Besides, the size of the NBA dataset varies between 4k and 20k and that of the Car dataset between 4k and 16k. Lastly, the missing rate in both datasets is fixed at 30% of the whole size of the dataset.

Figure 18(a) presents the results of the experiments conducted on the correlated synthetic dataset and demonstrate the relative error produced by our proposed approach and the previous approach. From the figure, we notice that our proposed approach produces less relative error rate between the real missing and the estimated values in comparison to *KNN*. This is due to the fact that our approach involves using the existing data in the database to capture the embedded relationship between the attributes in order to predicate the

missing values. Furthermore, due to the nature of the correlated data, it is relatively easy to determine the value of one attribute based on the value of the other attributes. It is also clear that *KNN* performs worse than our approach when the size of the database and the rate of the relative error increase with the size of the database in all cases. In contrast, the figure also shows that our approach maintains a very low relative error rate where increasing the size of the database has no significant impact on its performance. This improvement is caused by the positive effect of generating the *AFD* and assists in producing an estimated value that is very close to the real missing value.

In the independent dataset, Figure 18(b) illustrates that our proposed approach steadily outperforms the *KNN* when the size of the database increases. However, our approach performs slightly better as compared to *KNN* when the size of the database is 40k, and the improvement is slightly less compared to the correlated dataset. The main reason for this low improvement is caused by the fact that it is not easy to determine the embedded relationship between the attributes since the value of one attribute has no relation with the value of other attributes for independent data. However, we can observe that our approach produces an improved relative error rate in comparison to *KNN* when the size of the database falls within the range of 60 to 100k. We argue that the main cause of this significant improvement in the relative error of our approach is due to the fact that increasing the size of the database has a positive impact on the *AFD* generating process, which eventually helps provide a more precise estimation for the missing values.

Furthermore, Figure 18(c) explains the experimental results of the relative error rate for the NBA real dataset. It can be seen that the *AFD*-based approach shows some improvement over *KNN* when the size of the database is 4k and 12k. However, it can be noticed that our approach is not performing as well on the correlated dataset. Also, the figure depicts that there is no significant difference between the relative error rate produced in the *AFD*-based approach and *KNN* when the size of the database is greater than 16k. This insignificant improvement is due to the fact that the NBA dataset is considered as a semi-independent data which make it very challenging to obtain an accurate estimation for the missing values. Lastly, Figure 18(d) clarifies the results of the experiments that have been carried out on the Car real dataset. The figure suggests that our approach is superior as it steadily outperforms the previous approach in all cases. It is obvious that our approach is more scalable and performs better when the size of the database increases. In contrast, the figure shows that the performance of *KNN* declines and the relative error rate increases with larger database size. We thus conclude that our approach is superior to the other and also that increasing the size of the database has a positive influence on the performance of our approach.

It is clear from the experimental result presented in this section that the dataset size has a significant positive contribution to the relative error rate between the real missing value and the estimated value. This can be clearly seen in all cases of the experiments. However, the experimental result also shows that our proposed approach for value estimation of skylines can greatly benefit from the increment of the dataset size by producing a lower relative error rate between the real missing and estimated values. The main reason for this improvement in the value estimation using our proposed approach is due to the fact that our solution relies on exploiting the available data in the database when generating the estimated values. Thus, increasing the size of the database assists in capturing more implicit relationships between the attributes that lead to the generation of a larger number of AFDs and strengthens the relationship between the attributes. However, the results also demonstrate that the dataset size has no significant impact on improving the relative error rate between the real missing value and the estimated value for the *KNN* technique. This means, increasing the size of the dataset using *KNN* results in a lower relative error rate between the real missing and the predicted values in all cases of the experiments. Thus, the results presented in this section show that our proposed approach is superior for all cases in all experimental sets.

4) IMPACT OF THE USER-GIVEN THRESHOLD VALUE FOR THE ACCEPTABLE RELATIVE ERROR RATE

This set of experiments attempts to examine the impact of the user-given threshold value for the acceptable relative error between the real missing and the estimated values of the skylines. In this research work, we assume that the user can determine the acceptable estimated values if (and only if) the produced relative error rate between the real missing and the estimated values using local estimation is less than or equal to the user threshold value. It has to be noted here that the threshold value of the acceptable estimated values are given by the user (requester) when submitting the query to the incomplete database. Thus, if the relative error of the estimated values is greater than the user-given threshold value, these estimated values using our *AFD*-based approach for value estimation will be rejected followed by the attempt to estimate them using the crowd databases. We believe that the user-given threshold value for the acceptable estimated values using local estimation has a crucial impact on the amount of data to be outsourced from the crowd. Figures 19(a), 19(b), 19(c), and 19(d) present the experimental results of the outsourcing data size on the correlated, independent, NBA, and Car datasets, respectively. In this set of experiments, the details of the parameter setting of the correlated and independent datasets determine the size of the dataset to be 60k and the number of attributes including attributes with missing values to be six (6), while the rate of the missing value is fixed to 30%. Moreover, for the NBA and Car real datasets, the size of the dataset is fixed to 12k, while the number of attributes including attributes with missing values for the NBA is six (6) and four (4) for the Car datasets. Besides, the rate of the missing value for both real datasets is set to 30%. Lastly, the threshold value for the relative error rate of the acceptable

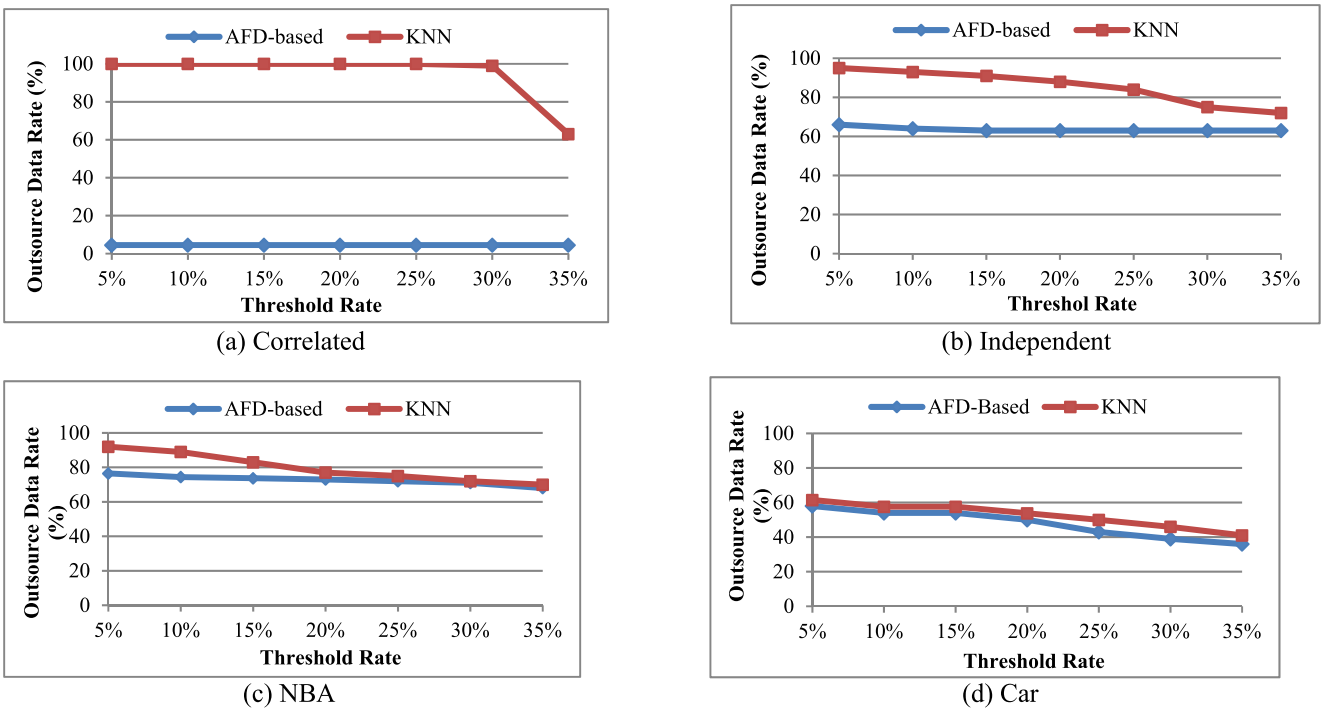

FIGURE 19. Impact of threshold value for the acceptable estimated missing values on the outsource data rate.

estimated values is varying between 5% and 35% for all datasets.

Figure 19(a) shows the results of the outsourced data size from the crowd for the missing values of the skylines over the correlated synthetic dataset. We observe here that our proposed approach is superior and steadily outperforms the previous approach in all cases. The figure also depicts that our *AFD*-based approach can provide an accurate estimation for the missing values event when the user-given threshold value for the acceptable estimated values is 5%. However, the results also show that the *KNN* is worse than the *AFD*-based approach and produces a large amount of missing data to be estimated using the crowd database. Nevertheless, the figure also suggests that the performance of *KNN* is slightly better in terms of the amount of missing data to be outsourced from the crowd when the user-given threshold value is set to 35%.

Figure 19(b) illustrates the results of the amount of data to be estimated using the crowd with respect to the user-given threshold value for the acceptable estimated values on the independent synthetic dataset. It suggests that the amount of missing skyline data to be estimated using the crowd based on our *AFD*-based approach is always less than those of the *KNN* approach in all cases. However, the amount of data to be estimated using the crowd database in the *KNN* approach is slightly lower when the threshold value is greater than 25%. Nevertheless, we can notice that the amount of missing data to be outsourced from the crowd using the *AFD*-based approach is much larger for the independent dataset than the amount of missing data to be outsourced from the crowd database for the correlated dataset (Figure 19(a)). This is due to the fact that for independent data it is difficult to produce an accurate value estimation using the local available data in the incomplete database. Since there are no implicit relationships between the attributes that can help generate the AFD to impute the missing values. Thus, most of the missing values have to be estimated using the crowd.

Figure 19(c) presents the results of the outsource data rate for the developed approaches on the NBA real dataset. From the figure, we can conclude that our *AFD*-based approach performs better than *KNN* in terms of the amount of data to be outsourced when the user-given threshold value is less than 20%. However, the performance of the *AFD*-based approach is slightly better than that of *KNN* in terms of the outsource data rate when the threshold rate of the acceptable estimated values is greater than 25%. From the figure, it is clear that the results of our approach over the synthetic datasets (correlated and independent) are superior to those produced by the NBA dataset as the NBA real dataset has a smaller data size than the correlated and independent dataset, which makes it difficult to identify the implicit relationships between the attributes and may result in a less accurate estimation of the missing skyline values. Most importantly, the NBA dataset is considered to be a semi-independent dataset, which renders the process of local estimation much more complicated and less accurate.

Finally, Figure 19(d) presents the experimental results on the Car real dataset that shows the amount of missing data to be outsourced from the crowd using both approaches.

The figure demonstrates that our approach outperforms the previous approach in terms of the amount of missing data to be outsourced from the crowd for all cases. We can also observe that there is a high relationship between the amount of missing data to be outsourced and the user-given threshold value for the acceptable estimated values. However, our approach is scalable and maintains a lower rate for the data to be outsourced from the crowd compared to the *KNN* approach.

B. EXPERIMENT RESULTS OF SKYLINES VALUE ESTIMATION USING CROWD-SOURCED DATABASES

1) THE MONETARY COST OF CROWD-SOURCED VALUE ESTIMATION

The monetary cost of the query result is considered as the most critical factor influencing the query processing in the crowd-sourcing databases. Besides, the monetary cost might also impact on the result accuracy of estimating the missing values in the database using the crowd. It has been argued that there is a tradeoff between the expected monetary cost and the result accuracy of the user given query [4], [5], [44]. This is also applicable when estimating the missing skyline values by exploiting the crowd resources including the available data in the crowd-sourced databases and the workers. Therefore, this set of experiments emphasizes on studying the relationship between the expected monetary cost of estimating the missing values of the skylines and the accuracy of the estimated values using the crowd. We have seen in the previous set of experiments that the relative error rate of certain value estimations of the skylines is lower than the user-given threshold value of the acceptable relative error rate between the real missing and the estimated values. Hence, we try to obtain a more precise estimation for these missing values of the skylines using the crowd-sourced databases. In this research work, the crowd-sourcing platform AMT (Amazon Mechanical Turk) is used to generate the estimated values of the missing values of the skylines. The monetary cost of estimating the missing values of the skyline using the crowd is computed as follows: First, we assume that the cost of estimating each missing value of the skylines through the crowd is \$0.05 per 1 judgment. Then the following monetary cost can be formulated:

$$Monetary\ Cost(MC) = EC^{*}No\_of\_Judgment^{*}Total\_no\_of\_MV$$

where:

EC: Expected Cost of estimation per missing value

No_of_Judgment: Number of Judgments per each missing value

Total_no_of_MV: Total number of Missing Values to be estimated from the crowd.

This experiment is aimed at examining the effect of the desired accuracy of the estimated values of the skylines using the crowd on the monetary cost; hence, the desired accuracy of the estimated values varies between 5% and 30% for each missing value in the skylines of the incomplete database.

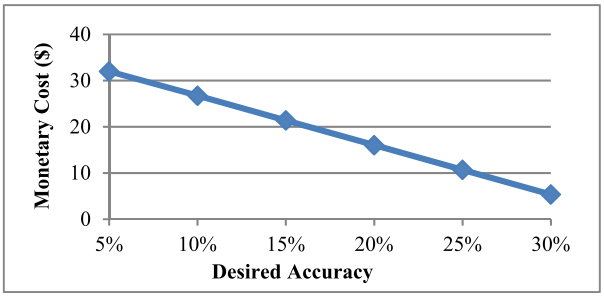

FIGURE 20. The effect of the desired accuracy of the estimated values on the monetary cost.

Figure 20 depicts the experimental results for the NBA real dataset in which the size of the dataset is fixed to 12k, the number of attributes including attributes with missing values is set to six (6), and the missing rate in the database is fixed to 30%. Here the total number of missing values to be estimated using the crowd databases is 107. From the figure, we can conclude that the total monetary cost of estimating the missing values of the skylines is highly influenced by the desired accuracy. We notice that when the desired accuracy is set to be 5%, the total monetary cost amounts to \$30; similarly, when the desired reduces accuracy the monetary cost is decreased. This is clear for the case when the desired accuracy is set to 10%, 15%, 20%, 25%, and 30%, the monetary cost is reduced to \$25, \$20, \$15, \$10, and \$5 respectively.

From the experimental results detailed in this section it can be noticed that the number of missing values to be estimated from the crowd and the desired accuracy of the missing values of the skylines have a significant impact on the monetary cost. Therefore, we can conclude that our proposed approach for value estimation of the missing values of the skylines presented in this research work is very beneficial since our *AFD*-based approach provides an accurate estimation for the majority of the missing values of the skylines without referring to the crowd. Hence, this process assists in reducing the number of missing values to be estimated from the crowd and leads to cheaper cost.

2) THE TIME LATENCY OF CROWD-SOURCED VALUE ESTIMATION

The main purpose of this set of experiments is to investigate the impact of the number of batches on the time latency to estimate the missing values of the skylines using the crowd [4], [5], [31], [40], [44]. The time latency indicates the total waiting time before retrieving the estimated values of the missing values of the skylines to the user. In this experiment, we assume that the missing values of the skylines to be estimated using the crowd will be delivered in batches that will be processed in parallel and the number of batches will vary between one (1) and six (6) batches. The NBA real dataset is used in this experiment, and the parameters setting of this experiment are the same as in the experiment reported in Figure 20. The crowd-sourcing platform AMT (Amazon Mechanical Turk) is used in this set of experiments as the

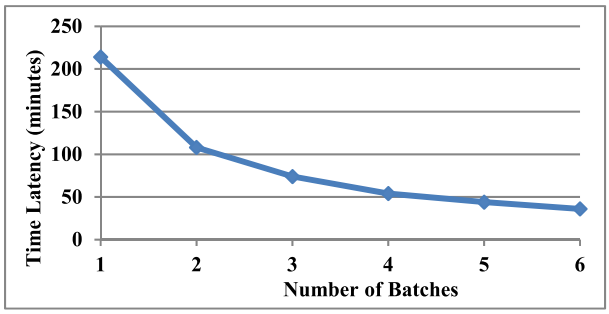

FIGURE 21. The crowd estimated time latency.

preferred platform for outsourcing the missing values of the skylines.

Figure 21 presents the results of the time latency to estimate the missing values of the skylines with respect to the number of batches to be returned. From the figure, we can observe that the time latency is highly influenced by the number of batches, whereby the number of batches increases with decreased time latency. We notice that given the number of batches as one (1) the time latency amounts to over 220 minutes. This trend can be seen for all other cases when the number of batches is 2, 3, 4, 5, and 6, the corresponding time latency becomes 105, 70, 52, 48, and 45 minutes respectively. Thus, a larger number of batches is very beneficial and leads to a significant decrement in the time latency to retrieve the estimated skyline values. Most importantly, reducing the number of values to be estimated from the crowd has a significant impact on the total time latency. The proposed approach of value estimation presented in this research work aims at minimizing the number of missing values to be estimated by the crowd, which in turn results in a decreased time latency of value estimation from the crowd.

VI. CONCLUSION AND FUTURE WORK DIRECTIONS

This paper addresses the issue of computing skyline queries over crowd-sourced enabled incomplete databases. An approach for skyline queries computation and missing values estimation of skylines on a database with incomplete data is proposed. The proposed approach provides a novel mechanism that is able to derive the most candidate skylines with incomplete data and thus reduce the amount of data involved in estimating the missing values of the skylines. To achieve this aim our approach entails applying data filtration that helps eliminate the unwanted tuples before applying the value estimation process. A new *AFD*-based strategy that provides an accurate estimation for the missing values based on the embedded relationships between the attributes of the incomplete database is proposed. The idea of utilizing the AFD allows to impute the missing values of the skylines by exploiting the local available data in the incomplete database and is capable of deriving skylines with no missing values in incomplete database systems. Lastly, our proposed approach successfully reduces the number of missing data to be estimated using the crowd without compromising the quality of the estimated values. In consequence, it helps minimize the monetary cost and reduces the time latency while maintaining high accuracy in the estimated values of the skylines using the crowd-sourced databases.

Future works can be conducted in the following directions. First, further studies can be undertaken to investigate processing top-k queries in crowd-sourced enabled incomplete database. It is quite challenging to reduce the monetary cost for top-k preference queries. This is due to the fact that this type of query requires scanning the entire database and thus involves complex processing in order to generate the final query result. For this type of query, a monotone function has to be formed that aggregate the values of all attributes and sorts the result in some order. Second, a further investigation can be accomplished to examine the effectiveness and the efficiency of the proposed approach against other machine learning-based techniques such as neural networks and generative adversarial network (GAN) for value estimation [1]. Third, another interesting area which can also be explored is dealing with uncertain data when processing skyline queries in crowd-sourcing databases. Issues such as monetary cost, time latency, and quality are highly affected by the uncertainty of the data. Last but not least, there are numerous research opportunities relevant to the issue of processing skyline queries on big data with incomplete data. In the big data context, the data are extremely diverse and grow rapidly, which increases the possibility of missing values. Working on big data requires powerful computing machines and extensive human resources in order to carry out a given task.

REFERENCES

[1] J. Lin, N. Li, M. A. Alam, and Y. Ma, "Data-driven missing data imputation in cluster monitoring system based on deep neural network," *Appl. Intell.*, vol. 50, no. 3, pp. 860–877, 2020.

[2] R. Wu, A. Zhang, I. F. Ilyas, and T. Rekatsinas, "Attention-based learning for missing data imputation in HoloClean," in *Proc. 3rd Conf. Mach. Learn. Syst. (MLSys)*, Austin, TX, USA, 2020, pp. 307–325.

[3] L. Beretta and A. Santaniello, "Nearest neighbor imputation algorithms: A critical evaluation," *BMC Med. Informat. Decis. Making*, vol. 16, no. 3, pp. 198–208, Jul. 2016.

[4] K. El Maarry, C. Lofi, and W. T. Balke, "Crowdsourcing for query processing on Web data: A case study on the skyline operator," *J. Comput. Inf. Technol.*, vol. 23, no. 1, pp. 43–60, 2015.

[5] C. Lofi, K. El Maarry, and W.-T. Balke, "Skyline queries in crowd-enabled databases," in *Proc. 16th Int. Conf. Extending Database Technol. (EDBT)*, Genoa, Italy, 2013, pp. 465–476.

[6] C. Lofi, M. K. El, and W. T. Balke, "Skyline queries over incomplete data-error models for focused crowd-sourcing," in *Proc. 32th Int. Conf. Conceptual Modeling (ER)*, Hong Kong, 2013, pp. 298–312.

[7] 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," *Int. J. Very Large Data Bases*, vol. 18, no. 5, pp. 1167–1190, Oct. 2009.

[8] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "CrowdDB: Answering queries with crowdsourcing," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, Athens, Greece, 2011, pp. 61–72.

[9] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2296–2319, Sep. 2016.

[10] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom, "CrowdScreen: Algorithms for filtering data with humans," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, Scottsdale, AZ, USA, 2012, pp. 361–372.

[11] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: A survey," *VLDB J.*, vol. 29, no. 1, pp. 217–250, Jan. 2020.
[12] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller, "Crowdsourced databases: Query processing with people," in *Proc. 5th Biennial Conf. Innov. Data Syst. Res. (CIDR)*, Asilomar, CA, USA, 2011, pp. 1–5.
[13] H. Park, R. Pang, A. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom, "Deco: A system for declarative crowdsourcing," *Int. J. Very Large Data Bases*, vol. 5. no. 12, pp. 1990–1993, 2012.
[14] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, Cancun, Mexico, Apr. 2001, pp. 421–430.
[15] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proc. 19th Int. Conf. Data Eng.*, Bangalore, India, 2003, pp. 717–719.
[16] 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)*, San Diego, CA, USA, 2003, pp. 467–478.
[17] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *Proc. 31st Int. Conf. Very Large Data Bases*, Trondheim, Norway, 2005, pp. 1–12.
[18] X. Lin, J. Xu, and H. Hu, "Range-based skyline queries in mobile environments," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 835–849, Apr. 2013.
[19] Y. Wang, B. Song, J. Wang, L. Zhang, and L. Wang, "Geometry-based distributed spatial skyline queries in wireless sensor networks," *Sensors*, vol. 16., no. 4, pp. 454–476, 2016.
[20] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for incomplete data," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Cancun, Mexico, Apr. 2008, pp. 556–565.
[21] 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.
[22] Y. Gulzar, A. A. Alwan, R. M. Abdullah, Q. Xin, and M. B. Swidan, "SCSA: Evaluating skyline queries in incomplete data," *Appl. Intell.*, vol. 49, pp. 1636–1657, Dec. 2018.
[23] Y. Gulzar, A. A. Alwan, and S. Turaev, "Optimizing skyline query processing in incomplete data," *IEEE Access*, vol. 7, pp. 178121–178138, 2019.
[24] 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, Aug. 2016.
[25] 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.
[26] R. Bharuka and P. S. Kumar, "Finding skylines for incomplete data," in *Proc. 24th Australas. Database Conf.*, Adelaide, SA, Australia, vol. 137, 2013, pp. 109–117.
[27] R. Bharuka and P. S. Kumar, "Finding superior skyline points from incomplete data," in *Proc. 19th Int. Conf. Manage. Data*, Ahmedabad, India, 2013, pp. 35–44.
[28] Y. Gao, X. Miao, H. Cui, G. Chen, and Q. Li, "Processing K-skyband, constrained skyline, and group-by skyline queries on incomplete data," *Expert Syst. Appl.*, vol. 41, no. 10, pp. 4959–4974, Aug. 2014.
[29] Y. Wang, Z. Shi, J. Wang, L. Sun, and B. Song, "Skyline preference query based on massive and incomplete dataset," *IEEE Access*, vol. 5, pp. 3183–3192, 2017.
[30] K. Zhang, H. Gao, H. Wang, and J. Li, "ISSA: Efficient skyline computation for incomplete data," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, Suzhou, China, 2016, pp. 321–328.
[31] A. A. Alwan, H. Ibrahim, and N. I. Udzir, and F. Sidi, "Missing values estimation for skylines in incomplete database," *Int. Arab J. Inf. Technol.*, vol. 15, no. 1, pp. 66–75, 2018.
[32] J. Lee, D. Lee, and S. W. Kim, "CrowdSky: Skyline computation with crowdsourcing," in *Proc. 19th Int. Conf. Extending Database Technol. (EDBT)*, Bordeaux, France, 2016, pp. 125–136.
[33] M. A. Soliman, I. F. Ilyas, and S. Ben-David, "Supporting ranking queries on uncertain and incomplete data," *Int. J. Very Large Data Bases*, vol. 19, no. 4, pp. 477–501, 2010.
[34] Y. He and D. C. Pi, "Improving KNN method based on reduced relational grade for microarray missing values imputation," *IAENG Int. J. Comput. Sci.*, vol. 43, no. 3, pp. 1–7, 2016.
[35] 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 (iiWAS)*, Yogyakarta, Indonesia, 2018, pp. 190–199.
[36] 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.
[37] Y. Gulzar, A. A. A. Aljuboori, N. Salleh, and I. F. Al Shaikhli, "Identifying skylines in cloud databases with incomplete data," *J. Inf. Commun. Technol.*, vol. 18, no. 1, pp. 19–34, Jan. 2019.
[38] Y. Zeng, K. Li, S. Yu, Y. Zhou, and K. Li, "Parallel and progressive approaches for skyline query over probabilistic incomplete database," *IEEE Access*, vol. 6, pp. 13289–13301, 2018.
[39] K. Zhang, H. Gao, X. Han, Z. Cai, and J. Li, "Modeling and computing probabilistic skyline on incomplete data," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 7, pp. 1405–1418, Jul. 2020.
[40] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2013, pp. 229–240.
[41] 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.*, early access, Oct. 11, 2019, doi: 10.1109/TKDE.2019.2946798.
[42] L. Caruccio, V. Deufemia, F. Naumann, and G. Polese, "Discovering relaxed functional dependencies based on multi-attribute dominance," *IEEE Trans. Knowl. Data Eng.*, early access, Jan. 17, 2020, doi: 10.1109/TKDE.2020.2967722.
[43] C. Luo, Z. Jiang, W.-C. Hou, S. He, and Q. Zhu, "A sampling approach for skyline query cardinality estimation," *Knowl. Inf. Syst.*, vol. 32, no. 2, pp. 281–301, Aug. 2012.
[44] J. Fan, M. Zhang, S. Kok, M. Lu, and B. C. Ooi, "CrowdOp: Query optimization for declarative crowdsourcing systems," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 8, pp. 2078–2092, Aug. 2015.
[45] G. Li, H. Yuan, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, and X. Zhang, "CDB: A crowd-powered database system," *Int. J. Very Large Data Bases*, vol. 11, no. 12, pp. 1926–1929, Aug. 2018.
[46] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar, "Answering enumeration queries with the crowd," *Commun. ACM*, vol. 59, no. 1, pp. 118–127, Dec. 2015.
[47] P. Cheng, X. Lian, X. Jian, and L. Chen, "FROG: A fast and reliable crowdsourcing framework," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 894–908, May 2019.
[48] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: Appropriate use and interpretation," *Anesthesia Analgesia*, vol. 126, no. 5, pp. 1763–1768, May 2018.

MARWA B. SWIDAN received the B.Sc. and M.Sc. degrees in computer science from the University of Tripoli, Libya, in 2004 and 2009, respectively. She is currently pursuing the Ph.D. degree with the International Islamic University Malaysia (IIUM), Malaysia. Her research interests include crowdsourcing, data mining/exploration, database systems, and skyline queries.

ALI A. ALWAN received the Master of Computer Science degree and the Ph.D. degree in computer science from Universiti Putra Malaysia (UPM), Malaysia, in 2009 and 2013, respectively. He is currently an Assistant 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 (LBSN), recommendation systems, and data management in cloud computing.

SHERZOD TURAEV received the Ph.D. degree in mathematics from the National University of Uzbekistan, in 2001, and the Ph.D. degree in computer science from Rovira i Virgili University, Spain, in 2010. He was a Postgraduate Researcher with University Putra Malaysia, from 2009 to 2012. He was an Assistant Professor with the Faculty of Information and Communication Technology, International Islamic University Malaysia, from 2012 to 2018. He was also an Associate Professor with the Faculty of Natural Sciences and Engineering, International University of Sarajevo, Bosnia and Herzegovina, from 2018 to 2019. He is currently an Associate Professor with the College of Information Technology, United Arab Emirates University. His research interests include graph theory, formal languages and automata, bio-computing, and information security. He serves as an editorial board member, a programming committee member, and a Reviewer with many international journals, such as *Mathematical Reviews*, *Theoretical Computer Science*, *Sains Malaysiana*, *Applied Mathematics and Computational Intelligence*, and many international conferences.

HAMIDAH IBRAHIM 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, data management in grid, and knowledge-based systems.

ABEDALLAH ZAID ABUALKISHIK received the master's and Ph.D. degrees in software engineering. He is passionate about the managerial process of software development, coding themes, big data, block chain, and data science. He is currently an Assistant Professor with the College of Computer Information Technology, American University in the Emirates, Dubai, UAE. He is also a Consultant with Regional Software Development Company, to prepare accurate estimation for project deliverables. He has published several high reputable refereed papers in high-impact factor journals and international conferences. His research interests include software functional size measurement, software functional measures conversion, cost estimation, empirical software engineering, database, big data, and data science. He is serving the Scientific Community as a Regular Reviewer for several high-impact journals.

YONIS GULZAR received the master's degree in computer science from Bangalore University, India, in 2013, and the Ph.D. degree in computer science from International Islamic University Malaysia, in 2018. He was a Part-Time Lecturer, a Teaching Assistant, and a Research Assistant with the Department of Computer Science, International Islamic University Malaysia. He is currently an Assistant Professor with King Faisal University (KFU), Saudi Arabia. His research interests include preference queries, skyline queries, probabilistic and uncertain databases, query processing, and optimisation and management of incomplete data, data integration, location-based social networks (LBSN), recommendation systems, and data management in cloud computing.