Skyline Queries over Incomplete Data - Error Models for Focused Crowd-Sourcing

Christoph Lofi[1], Kinda El Maarry[2], Wolf-Tilo Balke[2]

[1]National Institute of Informatics
Tokyo 101-8430, Japan
lofi@nii.ac.jp

[2]Institut für Informationssysteme
Technische Universität Braunschweig, 38106 Braunschweig, Germany
{elmaarry, balke}@ifis.cs.tu-bs.de

Abstract. Skyline queries are a well-known technique for explorative retrieval, multi-objective optimization problems, and personalization tasks in databases. They are widely acclaimed for their intuitive query formulation mechanisms. However, when operating on incomplete datasets, skyline query processing is severely hampered and often has to resort to error-prone heuristics. Unfortunately, incomplete datasets are a frequent phenomenon due to widespread use of automated information extraction and aggregation. In this paper, we evaluate and compare various established heuristics for adapting skylines to incomplete datasets, focusing specifically on the error they impose on the skyline result. Building upon these results, we argue for improving the skyline result quality by employing crowd-enabled databases. This allows dynamic outsourcing of some database operators to human workers, therefore enabling the elicitation of missing values during runtime. Unfortunately, each crowd-sourcing operation will result in monetary and query runtime costs. Therefore, our main contribution is introducing a sophisticated error model, allowing us to specifically concentrate on those tuples that are highly likely to be error-prone, while relying on established heuristics for safer tuples. This technique of focused crowd-sourcing allows us to strike a perfect balance between costs and result's quality.

Keywords: Skyline Queries; Error Models; Missing Data; Crowd-Sourcing;

1 Introduction

For the last decade, skyline queries have been a popular approach for personalizing database queries. By simply providing attribute preferences, users can quickly and intuitively obtain the best items of a dataset. However, skyline queries struggle with incomplete data, a common deficiency found regularly in real world datasets. Incomplete datasets missing some values interfere with the core concept of skyline queries, the *Pareto dominance,* which basically tests two database tuples, checking whether one of them is better than or equal to the other with respect to all attribute values (if it is, the second tuple can be safely excluded from the result as it is not one of the best tuples). This test cannot be performed reliably unless all the attribute values are known.

Despite its significant real world importance, the problem of incomplete datasets received only little attention by previous research on skyline computation. Usually, heuristics are used to decide whether a tuple a dominates a tuple b when those tuples exhibit missing attribute values. These heuristics either assume some default value for the missing attribute, or slightly alter the definition of Pareto dominance. However, the focus of these heuristics is mainly on efficient computation. Their actual *quality* with respect to the "correct" Skyline set − resulting from a respective dataset without missing information − has not yet been investigated. This shortcoming will be rectified in this paper through an extensive study, relying on several real world datasets. Our results will indicate that some of the heuristics that have been commonly used for skyline computations on incomplete data induce low quality results with many false positives or false negatives, while others fare significantly better.

Building on the insights obtained in this study, we will further improve the result quality by *selectively crowd-sourcing* some tuples, i.e. completing their missing values by having human workers retrieve the real values. Here, we rely on the capabilities of *crowd-enabled databases*[1] which render this process transparent and efficient. However, every crowd-sourcing operation will incur additional costs in terms of query time and money. Therefore, we investigate a hybrid approach in which we rely on using one of the proven heuristics surveyed in our study for some tuples, and crowd-source the others. For deciding if a tuple should be handled heuristically, we introduce an *error model* reflecting those tuples that are more likely to be correctly handled by the heuristics, and those that won't. Only those tuples for which the heuristic will likely fail are crowd-sourced, i.e. those tuples negatively impacting the correctness of the final result. By using this hybrid approach, we can find a good trade-off between result's correctness and query costs. Therefore, the contributions in this paper can be summarized as:

- Using real-word datasets, we *present* and *evaluate* common heuristics for dealing with skyline computation on incomplete data. Here, we focus especially on skyline result correctness and quality.
- We present an *error model* for handling tuples heuristically, leading to a *hybrid approach* for efficiently combining skyline heuristics and crowd sourcing.
- We extensively *evaluate* the quality and costs of our model and show that, with just small amounts of money, the result's correctness can be significantly improved.

2 Skyline Semantics and Skylines over Incomplete Data

In this section, we will cover the common heuristics for dealing with missing data in skyline computation, as well as the more recent ISkyline semantics [2]. Those previous works focused on computation efficiency or on reducing the size of the result set. The actual *error* induced by these heuristics compared to a real skyline computed from a complete dataset was never in the focus of attention. This issue is rectified by the survey carried out in this section, which also aims at finding a suitable baseline heuristic to be used for further improvement.

2.1 Formalizing Skyline Semantics

Skyline Queries [3] are a popular personalization technique for databases, successfully bridging set-based SQL queries and top-k style ranking queries [4]. They implement the concept of *Pareto optimality* from economics and thus allow for intuitive and simple personalization: for each relevant attribute users simply provide a preference order assuming ceteris-paribus semantics (e.g., "lower prices are preferred to higher prices given that all other attributes are equal"). Then, for any two tuples, where one tuple is preferred regarding one or more attribute(s) but equal with respect to the remaining attribute(s), rational users will always prefer the first object over the second one (the first object *Pareto dominates* the second one). Thus, the skyline set is computed by retrieving only tuples that are not dominated by any other tuple, i.e. all Pareto optimal tuples. In the basic case, skylines are computed on a complete dataset *R* without missing values:

Definition 1a (Complete Dataset): Formally, a dataset R is an instance of a database relation $R \subseteq D_1 \times \ldots \times D_n$ on n attributes $A_1, \ldots, A_n$ with D_i as domain of attribute A_i. Each tuple t is denoted by $t := (t_1, \ldots, t_m)$. For simplicity and without loss of generality, in the rest of this paper we only consider *score values* with respect to preferences, i.e. numerical domains and linearized categorical preferences normalized to $[0,1]$.

Now assume a user is stating a *skyline query*. Such a query is given by any set of preferences over the attributes of the base dataset.

Definition 2 (Numerical Preferences): A numerical preference P_i over attribute A_i with a numerical domain D_i is a *total order* over D_i. If attribute value $a \in D_i$ is preferred over value $b \in D_i$, then $(a, b) \in P_i$, also written as $a >_i b$ ("a dominates b wrt. to P_i"). Analogously, we define $a \gtrsim_i b$ for $a >_i b$ or $a =_i b$. Without loss of generality, we consider only *maximum score* preferences (i.e. all score values are in $[0,1]$, with 0 as worst and 1 as most preferred score).

Based on the preferences, Pareto dominance can be defined as:

Definition 3a (Pareto Dominance): We define the concept of *Pareto dominance* $t_1 >_P t_2$ between tuples $t_1, t_2 \in D_1 \times \ldots \times D_n$ by t_1 dominates or is equal to t_2 with respect to *all* attributes, and t_1 dominates $t_2$with respect to at least one attribute:

$$t_1 >_P t_2 \Leftrightarrow \forall\, i \in \{1, \ldots, n\}\colon \mathrm{t}_1 \gtrsim_i \mathrm{t}_2 \;\wedge\; \exists i \in \{1, \ldots n\}\colon \mathrm{t}_1 >_i \mathrm{t}_2$$

A *skyline query* is given by a set of preferences $P = \{P_1, \ldots, P_n\}$, with one preference specified for each attribute. Finally, the skyline of a dataset R is defined by:

Definition 4 (Skyline): On the complete dataset R and a set of preferences P, the skyline sky is defined as:

$$sky := skyline(R, P) = \{t_1 \in R \mid \nexists\, t_2 \in R : t_2 >_P t_1\}$$

Computing skyline sets is considered an expensive operation. Therefore, a variety of algorithms have been designed to significantly push the computation performance [5], e.g., by presorting [6], partitioning [7], parallelization [8], or multi-scanning the database [9–11].

2.2 Missing Data & Skylines

Missing data and incomplete datasets are becoming more and more common in modern information systems. This unfortunate development can mostly be attributed to automatically generated or aggregated datasets. The rise of Linked Open Data [12] contributes especially to this problem, where most LOD sources rely on error prone automated web scraping. Also, shopping portals and large e-commerce systems struggle hard to obtain complete datasets with all the relevant meta-data for a given product category.

Unfortunately, incomplete datasets with missing values pose a severe challenge for the original skyline semantics. When encountering a missing value during skyline computation, this basically means the test for Pareto dominance between two tuples $t_1 >_P t_2$ as given in definition 3 cannot be performed, and hence no meaningful skyline can be computed. Therefore, we will present common heuristics, which can deal with this issue in the next section. In the following, we will discuss the effects of incomplete data on skyline computation. We denote an incomplete (i.e. missing or unknown) value as □. This leads to the following definition for incomplete datasets:

Definition 1b (Incomplete Dataset): An incomplete dataset $R^{\square}$ is an instance of a database relation $R \subseteq D_1^{\square} \times \ldots \times D_n^{\square}$ on n attributes $A_1, \ldots, A_n$ with $D_i^{\square}$ as domain of attribute A_i using □ to denote a missing value, i.e. $D_i^{\square} = D_i \cup \{\square\}$. Each tuple t is denoted by $t := (t_1, \ldots, t_m)$. For each tuple, at least one attribute value is known.

Similar to definition 1a, we use normalized *score values*.

The subset of all complete tuples R^C is given by $R^C := \{t \in R^{\square} \mid \forall\, i \in \{1, \ldots, m\}: t_i \neq \square\}$ and the incomplete tuples are denoted as $R^I := R^{\square} \backslash R^C$.

2.3 Basic Heuristics

For handling missing information in skyline computation, several commonly used basic heuristics are at hand, providing default answers for deciding a Pareto dominance test. These heuristics can be classified into two general categories, optimistic and pessimistic heuristics. Pessimistic heuristics assume that missing values actually mask inferior values, and incomplete tuples will rarely be part of the skyline. In contrast, optimistic heuristics assume that incomplete tuples might actually be very good, and thus often promote them to be part of the skyline to avoid missing out any potential good candidate. The basic heuristics covered in our study are:

- *Incompleteness as failure* (ignore incomplete tuples): This simple pessimistic heuristic just ignores all tuples with missing values. Therefore, incomplete tuples cannot dominate other tuples, nor can they be in the final result set. This heuristic is obviously quite crude, and will result in both false negatives (i.e. incomplete tuples which would have been in the skyline, if their real values were known, but are ignored by the heuristic) and false positives (i.e. complete tuples which are assumed to be in the skyline, but one of the incomplete tuples would have dominated it, if its real values were known).
- *Treat incompleteness as incomparable*: This conservative optimistic heuristic aims at minimizing false negatives, i.e. no tuple should be excluded from the skyline

result unless it is clearly dominated by another tuple. As the test for dominance cannot be performed when an incomplete tuple is involved, those tuples don't dominate any other tuples, and at the same time can't be dominated. Therefore, incomplete tuples end up being in the skyline as there is no reliable information available indicating that they should be excluded. This potentially leads to many false positives, but only rarely to false negatives.

- *Surrogate with maximal values* (optimistic surrogation): This optimistic heuristic's approach differs from the two previous heuristics. Instead of providing a default decision for the dominance test, it assumes the values of missing information, i.e. it simply surrogates every missing value with the best possible value (1.0 for normalized score values). Then, the usual Pareto dominance semantics are applied for computing the skyline. The rationale behind this heuristic is that missing values are simply not known, and in the best case, they might be maximal. This heuristic may also lead to both false positives and false negatives.
- *Surrogate with minimal values* (pessimistic surrogation): This heuristic is similar to assuming maximal values, but takes a more pessimistic approach, surrogating missing values with the minimal value (e.g., 0.0). This allows incomplete tuples to be in the skyline result, but only if the tuple shows superior values for at least one of the known attributes. Therefore, this heuristic will mostly induce false negatives (incomplete tuples which should be in the skyline, but are now dominated due to the assumption of minimal values for their missing attributes).
- *Surrogate with expected values* (value imputation): This approach relies on various statistical means to predict the expected values of incomplete tuples, i.e. missing values are replaced by their estimated "real" values. The efficiency of this heuristic on skyline queries has been covered in detail in [13]. In the following, k-nearest neighbor value (KNN) imputation [14] will be used as it has been shown to be one of the stronger value prediction heuristics for general real live datasets.

2.4 ISkyline and Weak Pareto Semantics

ISkyline semantics [2] are one of the latest works centrally dealing with heuristic skyline computation over incomplete data. They closely resemble Weak Pareto Dominance published earlier in [15] and [16]. Weak Pareto and ISkyline semantics (we refer to both as simply ISkyline in the following) change the actual semantics of Pareto dominance in order to respect missing values (see def. 3b): a tuple dominates other tuples, if it is better regarding at least one attribute and at least equal *or showing missing values* in all other attributes. However, this definition implies non-transitive dominance relationships and may lead to cyclic dominance behavior. Non-transitivity poses severe problems for traditional skyline algorithms, where relying on transitivity is one of the key techniques for implementing efficient skyline computation. Accordingly, ISkyline also provides an alternative skyline computation algorithm that deals with non-transitivity.

Therefore, while using ISkyline dominance, only those in common attributes whose values for both tuples are known are considered and then traditional dominance semantics are applied. Consequently, this heuristic can also lead to larger numbers of false positives and false negatives.

Definition 3b (ISkyline Dominance): The concept of *ISkyline dominance* $t_1 >_{IS} t_2$ between two tuples $t_1, t_2 \in D_1 \times ... \times D_n$ is given by t_1 dominates $t_2$with respect to at least one attribute for which no values are missing, and for all other attributes, t_1 dominates or is equal to t_2 or one or both attribute values are missing:

$$t_1 >_{IS} t_2 \Leftrightarrow$$
$$\forall\, i \in \{1, \dots, n\}: (t_1 = \square \vee\ t_2 = \square\ \vee t_1 \gtrsim_i t_2)\ \wedge\ \exists i \in \{1, \dots n\}: t_1 >_i t_2$$

2.5 Evaluation of Heuristics for Skylines on Incomplete Data

In this section, we will evaluate and compare the previously presented heuristics from a purely quality-focused point of view. This will allow us to select one heuristic that provides the highest quality results for further improvement in the second part of this paper. We basically compare the quality of a result derived from one of the heuristics with that obtained from the corresponding complete dataset.

Previous studies on skyline queries show a clear connection between skyline sizes and the degree of correlation in data [17, 18]. If the data is highly correlated, then skyline queries indeed reduce the result size drastically, fulfilling their promise of being a powerful and intuitive query personalization tool. However, if the data is anti-correlated, skyline results can easily contain 50% or even more of all database tuples. Therefore, if small skylines are to be expected, pessimistic heuristics will provide better results closer to the real results, while for anti-correlated data, optimistic approaches, which favors incomplete tuples in the skyline, will fare better.

In the upcoming evaluations, we will abstain from experimenting with synthetic data, and instead evaluate using three real-word e-commerce datasets for judging the heuristics under realistic circumstances. As with most real life datasets, our datasets also show a higher degree of correlation. All our datasets are complete, and values are artificially removed for the experiments:

a) Our first dataset is the well-known NBA player statistics (http://www.basketballreference.com). It consists of 21,961 tuples. For each player, we used 5 attributes, i.e. games played, points scored, rebounds, assists, and goals. We use maximum preferences (i.e. larger values are considered to be better than smaller values), resulting in a skyline of 75 tuples.

b) The second dataset contains 1,597 notebooks, crawled in 2010 from Dooyoo.de (http://www.dooyoo.de/notebook). This dataset features 6 attributes: CPU frequency (maximum preference), CPU type (categorical preference encoded by a score), RAM (max.), HD (max.), display size (max.), and weight (minimum preference), resulting in a skyline of 35 tuples.

c) The third dataset contains different car models, crawled from Heise.de (http://www.heise.de/autos/neuwagenkatalog) in 2011, including 7,755 tuples with the following attributes: price (min.), power (max.), acceleration (max.), fuel consumption (min.), CO_2 emission (min.), and taxes (min.). It results in a skyline of 268 tuples.

In order to compare and evaluate the five heuristics, values are removed randomly from the datasets, ranging from 1% (e.g. nearly-complete dataset) to 20%. So basically, we simulate incomplete datasets while retaining the complete dataset as a reference for error computation.

For measuring the actual error of a skyline set computed by a heuristic, we rely on the inverse of *Informedness* [19], a popular metric from information retrieval. Informedness quantifies how informed a computed result is when compared to a result derived by chance. The informedness measure is based on recall and inverse recall. In contrast to using recall alone, it considers error types, false positives and false negatives, while simultaneously taking into account true positives and true negatives. Therefore, it is a fair and unbiased measure.

Definition 5 (Skyline Error): Let sky_H be a skyline computed by a chosen heuristic applied to an incomplete dataset, and sky_R be the real skyline computed from the complete dataset. The error between both sets is given by (some arguments omitted):

$$error(sky_{\mathrm{H}}, sky_{\mathrm{R}}) = 1 - informdness(..)$$

$$informedness(sky_{\mathrm{H}}, sky_{\mathrm{R}}) = recall(..) + invRecall(..) - 1$$

$$recall(sky_{\mathrm{H}}, sky_{\mathrm{R}}) = \frac{truePositives(..)}{truePositives(..)+falseNegatives(..)}$$

$$invRecall(sky_{\mathrm{H}}, sky_{\mathrm{R}}) = \frac{trueNegatives(..)}{trueNegatives(..)+falsePositives(..)}$$

Focusing on the skyline error incurred by each heuristic as the missing values in the entire dataset increases from 1 to 20%, we can see a rather consistent result across the three datasets (see Figure 2). Using the ISkyline semantics [2] yields the highest skyline error, whilst using minimal value surrogation consistently results in the smallest skyline error, with the exception to the NBA dataset where surrogating with the KNN-predicted value fares better. However, this should be regarded as a special case attributed to the predictability and uniformity of that particular dataset. Also, maximal values surrogates show bad results (however, optimistic approaches are expected to be less effective for data with high correlation). The two basic heuristics that rely on default handling (ignoring incomplete tuples & incompleteness as incomparable) show similar middle ground results, even though one is optimistic and the other pessimistic. This can be attributed to their local nature which does not allow far-reaching consequences (e.g. despite being optimistic and including incomplete tuples in the skyline, the 'incomparable' heuristic does not allow incomplete tuples to dominate other tuples. Very much in contrast to the other optimistic maximum surrogation heuristic, which owes its bad results to complete tuples that has been wrongfully excluded.) Interestingly, the more complex KNN value imputation heuristic, which surrogates with expected values, only results in an average skyline quality.

Basically, this survey shows that from a pure quality perspective, the simple pessimistic approach surrogating all missing values with the minimal value yields the best results, rendering all other approaches (particularly the optimistic ones) vastly inferior when focusing on the closest resemblance with the correct result.

But still, while pessimistically surrogating with minimal values does lead to better results than all the other heuristics, it severely discriminates against incomplete tuples, which consequently have only now slim chances to be included in the result. This drawback is rectified by our hybrid skyline approach, which diminishes this imbalance and further improves the result's quality by obtaining additional information using crowdsourcing.

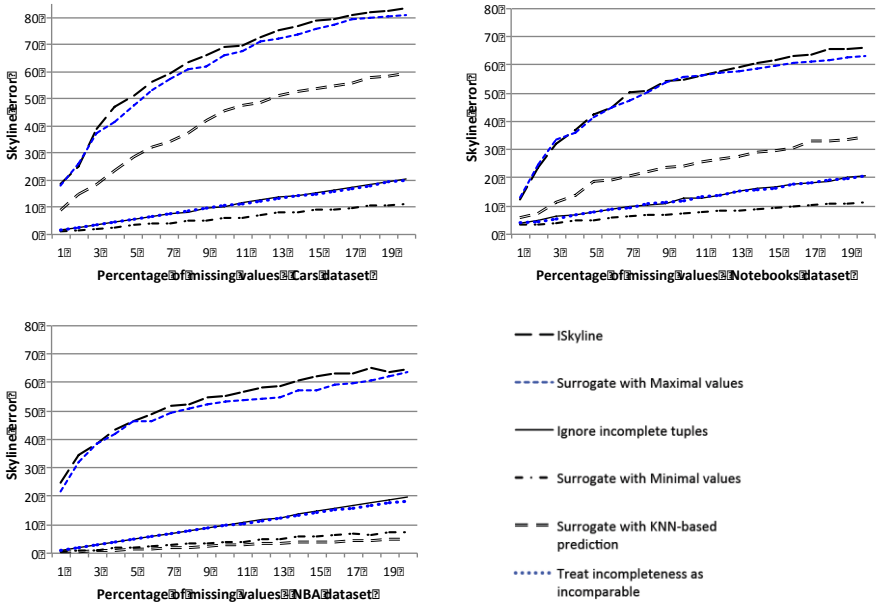

Figure 2: Skyline error at different % of missing values for various adapting skylines to incomplete datasets heuristics

3 Improving Skyline Quality with Focused Crowd-Sourcing

In this section, using the abilities provided by Crowd-enabled DBMS, we aim at improving the skyline result's quality that was achieved by heuristic approaches for missing data through crowd-sourcing some of the missing values in a focused fashion. Crowd-enabled DBMS [1] have proven to be a very powerful and popular technology, fusing traditional relational technology with the cognitive power of people. Here, the DBMS can issue operations during runtime in the form of Human Intelligence Tasks (HITs) to crowd-sourcing services like Amazon's Mechanical Turk, CrowdFlower, or SamaSource. This technique can be used to complete the tables' missing data in a *query-driven fashion*, i.e. queries can be executed by filling empty fields despite incomplete data. Moreover, even complex cognitive tasks like reference reconciliation can be performed, and the crowd can also be leveraged to "implement" database operators that would require cognitive abilities like scoring tuples with respect to perceived criteria (e.g., images by visual appeal), or performing perceptual comparisons (e.g., tagging images with emotions).

A major issue in crowd-sourcing is the reliability of the results obtained from the workers due to malicious or simply incompetent workers. In most cases, these concerns can be addressed effectively with quality control measures like majority voting or Gold sampling [20]. In previous studies on crowd-sourcing it has been shown that within certain bounds, missing values in database tuples can be elicited with reliable efficiency and quality as long as the information is generally available. That's especially true for

factual data that can be looked-up on the Web without requiring expert knowledge (e.g., product specifications, telephone numbers, addresses, etc.). In such a case, the expected data quality is high with only a moderate amount of quality assurance (e.g., majority votes). For example, [20] reports that crowd-sourced manual look-ups of movie genres in IMDB.com are correct in ~95% of all cases with costs of $0.03 per tuple (including quality assurance). Accordingly, further investigations into workers' quality are not a focus of this work.

Efficiency-wise, while each individual HIT might be cheap, costs can quickly sum up. Furthermore, each HIT requires some time for the human workers to complete the task. Because of this, the naïve approach, i.e. just crowd-sourcing all missing attribute values, is prohibitively expensive as most information that has been obtained with high costs will not even be part of the final result set.

Balancing the monetary cost and time against the desired results or improvements is therefore of utmost importance. Consequently, this section provides a hybrid approach that selectively crowd-source only the most relevant tuples, while relying on heuristics for all the others. The goal is to compute at minimal costs a skyline set that is as close as possible to the skyline which would've been obtained had all information been available (please note that the focus is on identifying the correct tuples and not on having a skyline result set without missing information).

3.1 Error Models for Focused Crowd-Sourcing for Skyline Queries

In order to achieve our goal of striking a favorable balance between low costs and high quality, we introduce and evaluate three ranking heuristics that rank all tuples with missing values with respect to their potential negative effects on the skyline. Next, we provide an error model for identifying only those tuples with the highest negative potential to be accordingly crowd-sourced. This enables us to tightly restrict the crowd-sourcing costs, and quickly reach at the same time a significantly better final result quality. Our resulting *two-stage approach* works as follows:

- Relying on the study results of the previous section, we surrogate all missing values with the minimal values as a baseline heuristic, i.e. all missing values are replaced by 0. This approach leads to a strong baseline in terms of quality even before crowd-sourcing. Furthermore, this heuristic has another valuable property: every tuple ending in the skyline's result set when surrogating with 0 has a very high probability to be a true positive (i.e. even if the real value is known, the tuple will most likely stay in the skyline). Therefore, there is no need after minimal value surrogation to crowd-source any tuple that ends up in the skyline's result.
- After the initial heuristic handling, the result's quality is improved b crowd-sourcing some of the tuples to obtain their real values. Here, we can safely focus on incomplete non-skyline tuples. Furthermore, we try to crowd-source only those tuples that are most likely to be false negatives and ignore all others (i.e. ignore those which are most likely true negatives). To ultimately decide which tuples to crowd-source, the following error models aim at capturing the likeliness that an incomplete tuple is indeed a false negative.

Error Model based on Potentially Dominated Tuples: In this model, we rely on counting the number of tuples a given tuple dominates when its missing values have been surrogated. All incomplete tuples are then ranked by this count, and the top tuples (i.e. those which potentially dominate most tuples) are assumed as being most error prone and therefore crowd-sourced. Every time a tuple is crowd-sourced, this ranking is recomputed to adapt to the changes of the new information and is then reflected upon the skyline's result set. Please note that this error model, used in the second stage of our process, is independent of the heuristic used in the first stage (that chosen heuristic is only applied to tuples considered safe by the error model). We studied two variants of this error model.

a) Min: In the first error model, we simply count the number of dominated tuples when surrogating all missing values with the minimal value (called *minimum model* in the following). Unfortunately, this approach is less effective (see evaluations), where only few or even no other tuples are usually dominated when surrogating with minimal values. Furthermore, this variant also ignores the possible potential of tuples, as usually most real values are better than 0 (i.e. it is too pessimistic and an optimistic approach would fare better for ranking).

b) Min-max: Therefore, as a second variant, we temporarily (i.e. only for ranking tuples) surrogate the missing values of the current tuple with the maximum value, while retaining the minimal surrogation for all the other tuples to be ranked. This heuristic is called *min-max model* (See definition 6). This optimistic ranking heuristic leads to significantly better results due to its higher discriminating power. After crowd-sourcing, the missing values of all non-crowd-sourced tuples are again reverted to minimal values (our baseline heuristic) for the final skyline computation (i.e. optimistic handling during ranking, pessimistic handling for skyline computation of non-crowd-sourced tuples).

Definition 6 (Minimal Maximal Replacement Error Model): For a dataset $R^{\square} = R^C \cup R^I$ containing complete and incomplete tuples with n attributes $A_1, \dots, A_n$, the number of potentially dominated tuples for a given tuple $t \in R^I$ can be computed by:

$$maxDomCount(t) = |\{t_d \in R^C | \hat{t} >_P t_d\}|$$

with

$$\hat{t}_i = \begin{cases} 1 : if\ t_i = \square \\ t_i : if\ t_i \neq \square \end{cases}$$

Error model based on Impact of Missing Tuples: Not all attributes have an equal impact on the skyline result, and some attributes can be more influential for deciding a tuple's membership in the skyline than others. This supposition is quite sensible – and as will be shown in the evaluation section valid – as it mimicks the typical and common importance a user may lay on attributes when making a decision. In this error model, the potential impact of all attributes is measured in an initial dataset analysis phase. Here, we focus on measuring the skyline error introduced when a given attribute is completely ignored. Using the subset of all complete tuples, we compute the skyline. Then, we iteratively ignore each attribute, treating it as completely absent and re-compute the skyline. Comparing both skylines and computing the skyline error based on the informedness measure (see Definition 5), we get the error this attribute is responsible for introducing into the skyline's result.

This impact measure can be then combined with the previous minimal-maximal replacement heuristic to provide a better ranking, consequently improving the skyline error furthermore while still retaining similar crowd-sourcing costs (as given by def. 7). This is achieved by ranking all tuples with respect to: *Number of Dominated Tuples * (sum of the attribute impact of all missing attribute values)*. It is important to note that a tuple missing multiple attributes don't necessarily score higher than tuples with fewer missing attributes. It depends on how big the associated error of a missing attribute is, and so the sum of two missing attributes can easily be smaller than that of one highly influential attribute.

Definition 7 (Attribute Impact): For a dataset $R^{\square} = R^{C} \cup R^{I}$ containing complete and incomplete tuples with n attributes $A_1, \dots, A_n$ and corresponding attribute impact error vector $(I_1, \dots, I_n)$, the total impact error I_t of a tuple $t \in R^I$ is given by

$$I_t := \sum_{x \in \{i | t_i = \square\}} I_x$$

Finally, all incomplete tuples $t \in R^I$ are ranked by their weighted minimal-maximal domination count:

$$weightedCount(t) = maxDomCount\ (t)\ \times\ I_t$$

3.2 Evaluation

Analogous to our previous evaluation in section 2.5, we now focus on evaluating how the skyline error can be further improved effectively by just few crowd-sourcing operations. First, in an initial dataset analysis phase, we measure the attributes' impact vector for each of our three datasets. Next we investigate which of the error models perform best (min value, min-max value, attribute impact with min-max value). Finally, we measure the efficiency of our approach in a real crowd-sourcing experiment.

Measuring the impact of missing attributes: In an initial dataset analysis phase on the three datasets, the following missing attribute impact's error values shown in table 1, 2 and 3 were obtained by measuring the introduced skyline's error when the corresponding attribute was ignored or completely missing. Some attributes instantly stand out, displaying a more influential role than the others. In the Notebooks dataset, *Weight*, *Display Type* and *CPU Frequency* have more sway on a tuple being in the skyline than the *CPU,* which merely impacts the skyline's quality by 8.632. Similarly in the Cars dataset the *Power* and *Price* should be carefully considered when missing. A tuple

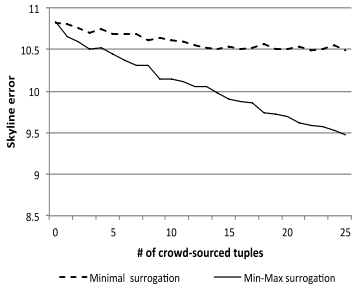

Figure 3: Skyline improvement comparing the two counting variants

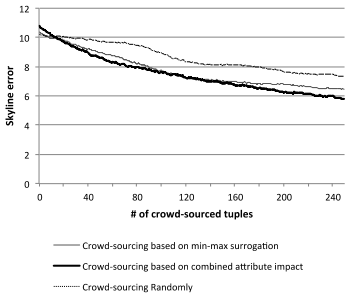

Figure 4: Min-max counting, impact heuristic, and random crowd-sourcing

missing the value for the *Power* attribute should be treated as more highly error-prone than a tuple missing the *C0²-Emission*.

Basic Error Model Evaluation: In this section, we evaluate our two-phase hybrid approach with both error models and crowd-sourcing.

In the first stage of our approach, we apply the heuristic surrogating with minimal values as a default handler (refer to section 2.3). For the second stage, i.e. when selecting tuples for crowd-sourcing, we focus only on tuples that could be potentially false negatives when handled heuristically. We employ our error models on these tuples to decide which of them will likely impact the result most negatively, and crowd-source these, but still handle all others heuristically. As it turns out, applying the minimal surrogation error model yields only miniscule skyline error reduction when compared to randomly crowd-sourcing tuples. Applying this model on the cars dataset with 20% missing values, the skyline error decreased from 10.8 to only 10.4 as depicted in Figure 3. This is because surrogating with minimal values is a bad heuristic for estimating the potential impact of a tuple; therefore the discriminative power of this model is rather low. Also, the model has the undesired effect that after crowd-sourcing 25 tuples, no further meaningful ranking is possible as none of the remaining incomplete tuples dominates any other tuples in the dataset.

Therefore, temporarily using maximal values as surrogates as in the min-max model clearly out-performs the minimal value surrogation variant, as the skyline error decreases from 10.8 to 9.8 instead of to only 10.4 for just 25 crowd-sourcing operations. Furthermore, compared to previous studies on crowd-sourcing for skylines in [13], both these approaches significantly outperform the focused crowd-sourcing with the KNN-predicted values, where the skyline error starts high at 61.8 and decreases to only 19.8 after 25 crowd-sourcing operations. Also, when using the min-max error model, a meaningful ranking for more than 25 tuples can be created, thus allowing the focused crowd-sourcing skyline to rise to its full potential with further tangible improvements to quality.

These results can be further improved by also considering the attribute impact, as illustrated in figure 4. Here, the skyline error drops from 10.8 to 5.7 upon crowd-sourcing 250 tuples out of 7,755 tuples (compared to crowd-sourcing 271 tuples when combined with min-max error models, and 358 tuples for random crowd-sourcing). Naturally, randomly crowd-sourcing just some tuples (i.e. without following any error

NBA Dataset	**Impact Error**
Games played	62.667
Points scored	2.667
Total rebounds	50.667
Assists	78.667
Field goals made	6.667

Table 1 (left): Attribute impact error analysis for NBA dataset

Notebooks Dataset	**Impact Error**
CPU	8.632
CPU Frequency	40.181
RAM	17.323
Hard Drive	25.835
Display Type	68.571
Weight	88.571

Table 2 (bottom-left): Attribute impact error analysis for Notebooks dataset

Cars Dataset	**Impact Error**
Price	78.534
Power	90.312
Acceleration	33.609
Fuel Consumption	28.758
CO^2 Emission	10.848
Taxes	47.775

Table 3 (bottom-right): Attribute impact error analysis for Cars dataset

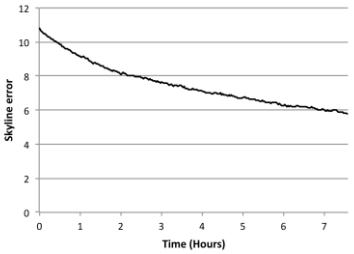

Figure 5: Time required for CS (Cars dataset)

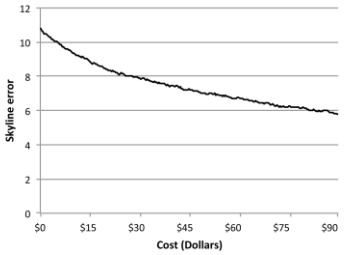

Figure 6: Cost for CS in dollars (Cars dataset)

model) leads to a very slow improvement of the result quality, and thus incurs higher costs and longer time to reach the same skyline quality achieved by either model.

Crowd-Sourcing's Costs & Time: In this last set of experiments, we evaluate the efficiency of our approach through a real crowd-sourcing experiment, focusing on the monetary and time costs. We used CrowdFlower.com as a crowdsourcing platform, and again chose the cars dataset with 20% of missing values to run our experiment. As described in 3.1, we started with a skyline based on minimal-surrogation heuristics, and then crowd-sourced one tuple at a time relying on min-max error models with attribute impact for ranking the tuples to be crowd-sourced. And then we measured the skyline error after each crowd-sourcing operation. In order to obtain reliable values from the crowd-workers, for every crowd-sourced value, a majority vote from 4 workers was required for quality control, i.e. each single value was crowd-sourced multiple times. Thus, due to this high overhead for guaranteeing high quality results, each value cost 0.36$ and took 1.8 minutes on average. Figure 5 and 6 illustrates the skyline result error improvement from 10.8% to 5%. In the end, crowd-sourcing 250 out of the 7,755 overall tuples roughly required 7.5 hours and 89$.

4 Summary & Outlook

In this paper, we extensively studied the effects of different heuristics for evaluating skyline queries on incomplete datasets with missing values. These studies used three real-life datasets, and different degrees of incompleteness have been considered. The results showed that surrogating missing values with the least desirable values shows the best results with respect to skyline correctness, while other popular approaches like treating missing values as being incomparable or the ISkyline semantics result in significantly lower skyline correctness. Building upon this insight, we developed additional error models that identify those tuples that will strongly degenerate the final result quality even when using good heuristic handling. In order to further improve result quality, we developed a hybrid approach where those tuples which severely impact quality as identified by the error model are selectively crowd-sourced to human workers in order to obtain their real values, while those tuples which are considered "safe" are handled by the respective skyline heuristic. This hybrid approach allows us to fine-tune a favorable trade-off between result quality and query costs, as just few crowd-sourced tuples can significantly improve the correctness of the skyline result.

References

1. Franklin, M., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: CrowdDB: Answering queries with crowdsourcing. ACM SIGMOD Int. Conf. on Management of Data. , Athens, Greece (2011).
2. Khalefa, M.E., Mokbel, M.F., Levandoski, J.J.: Skyline Query Processing for Incomplete Data. Int. Conf. on Data Engineering (ICDE). , Cancun, Mexico (2008).
3. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. Int. Conf. on Data Engineering (ICDE). , Heidelberg, Germany (2001).
4. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. Symposium on Principles of Database Systems (PODS). , Santa-Barbara, California, USA (2001).
5. Godfrey, P., Shipley, R., Gryz, J.: Algorithms and analyses for maximal vector computation. The VLDB Journal. 16, 5–28 (2007).
6. I. Bartolini, Ciaccia, P., Patella, M.: Efficient sort-based skyline evaluation. ACM Transactions on Database Systems. 33, (2008).
7. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30, 41–82 (2005).
8. Selke, J., Lofi, C., Balke, W.-T.: Highly Scalable Multiprocessing Algorithms for Preference-Based Database Retrieval. Int.Conf.on Database Systems for Advanced Applications (DASFAA). , Tsukuba, Japan (2010).
9. Torlone, R., Ciaccia, P.: Finding the best when it‘s a matter of preference. 10th Italian Symposium on Advanced Database Systems (SEBD). , Portoferraio, Italy (2002).
10. Boldi, P., Chierichetti, F., Vigna, S.: Pictures from Mongolia: Extracting the top elements from a partially ordered set. Theory of Computing Systems. 44, 269–288 (2009).
11. S. Park, Kim, T., Park, J., Kim, J., Im, H.: Parallel skyline computation on multicore architectures. Int.Conf. on Data Engineering (ICDE). , Shanghai, China (2009).
12. Heath, T., Hepp, M., Bizer, C. eds: Special Issue on Linked Data. International Journal on Semantic Web and Information Systems (IJSWIS). 5, (2009).
13. Lofi, C., Maarry, K. El, Balke, W.-T.: Skyline Queries in Crowd-Enabled Databases. Int. Conf. on Extending Database Technology (EDBT). , Genoa, Italy (2013).
14. Acu, E.: The treatment of missing values and its effect in the classifier accuracy. Classification clustering and data mining applications. 1–9 (2004).
15. Balke, W.-T., Güntzer, U., Siberski, W.: Exploiting Indifference for Customization of Partial Order Skylines. Int. DB Engineering & Applications Symposium (IDEAS). , Delhi, India (2006).
16. Balke, W.T., Güntzer, U., Siberski, W.: Restricting skyline sizes using weak Pareto dominance. Informatik - Forschung und Entwicklung. 21, 165–178 (2007).
17. Balke, W.-T., Zheng, J.X., Güntzer, U.: Approaching the Efficient Frontier: Cooperative Database Retrieval Using High-Dimensional Skylines. Int. Conf. on Database Systems for Advanced Applications (DASFAA). , Beijing, China (2005).
18. Godfrey, P.: Skyline cardinality for relational processing. How many vectors are maximal? Symp. on Foundations of Information and Knowledge Systems (FoIKS). , Vienna, Austria (2004).
19. Powers, D.M.W.: Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Flinders University Adelaide SIE07001. (2007).
20. Lofi, C., Selke, J., Balke, W.-T.: Information Extraction Meets Crowdsourcing: A Promising Couple. Datenbank-Spektrum. 12, (2012).