Crowd-Sourcing for Query Processing on Web Data: A Case Study on the Skyline Operator

Abstract: In recent years, crowdsourcing has become a powerful tool to bring human intelligence into information processing. This is especially important for Web data which in contrast to well-maintained databases is almost always incomplete and may be distributed over a variety of sources. Crowdsourcing allows to tackle many problems which are not yet attainable using machine-based algorithms alone: in particular, it allows to perform database operators on incomplete data as human workers can be used to provide values during runtime. As this can be become costly quickly, elaborate optimization is required. In this paper, we showcase how such optimizations can be performed for the popular skyline operator for preference queries. We present some heuristics-based approaches, and compare them to crowdsourcing-based approaches using sophisticated optimization techniques while especially focusing on result correctness.

1. Introduction

Crowdsourcing has become a popular approach to many problems on the Web that cannot be easily addressed by automated methods and algorithms, or problems that explicitly require significant amount of human intelligence or human feedback. Here, human workers are recruited to perform small tasks usually in exchange for monetary compensation. Crowdsourcing can be applied to a multitude of different problems from artificial intelligence to e-commerce. Here crowd-enabled data processing [1] demands a special mention as it tackles a frequently reoccurring problem in modern information systems: allowing to perform database operators on incomplete data. This is achieved by eliciting missing information directly from human workers during runtime. However, costs in both monetary compensation and time can quickly escalate, and efforts have to be made to optimize query executing to ensure that no data is expensively elicited if it is not needed for a query result. While some of these optimizations are quite generic, often specialized optimization techniques for each individual database operator are required. Therefore, in this paper, we showcase how such optimizations can be performed for the popular Skyline query operator.

These Skyline queries have always been a popular and promising approach to personalizing database queries. By simply providing attribute preferences, users can quickly and intuitively obtain the best items of a dataset with respect to these individual preferences. Most previous research assumes complete underlying datasets and focuses either on the computational costs of skyline queries´ execution, alternative semantics, or the size and manageability of the result sets. Unfortunately, the reality is harshly different. Nowadays, with the widespread use of automated information extraction and aggregation, incomplete datasets (i.e. datasets missing values for some tuples) have become a fairly common shortcoming. This is particularly pressing for skyline queries, as the incompleteness directly interferes with the core concept upon which they are based on, namely, the Pareto dominance (i.e. skyline queries filter out tuples that are inferior to others in all respects, retaining only “optimal” ones).

So far whenever the underlying dataset is incomplete, only simple heuristics are used to decide whether a tuple is dominated by another with respect to the Pareto semantics. Typically, such heuristics range from: assuming some default value for missing attributes, slightly altering the definition of Pareto dominance semantics, or to deciding the Pareto test with some default result.

Moreover, the main focus of these heuristics is again on efficient computation, and no particular attention is paid to the "quality" and "correctness" of the resulting skyline set. Therefore, in this paper, we discuss the challenging question of *how to run skyline queries on incomplete datasets while focusing on high result quality*. To address this question, we thoroughly present two different approaches.

- Advanced heuristic approach: In contrast to other heuristics, this heuristic approach focuses on the correctness of a skyline. Accordingly, the main aim is to compute a skyline result that resembles the skyline that would have been computed had the underlying dataset been complete as closely as possible. Typically, the correctness of a skyline degrades when relying on heuristics for deciding Pareto dominance as some tuples will be: 1) wrongfully included in the skyline result (false positives), or 2) left out when they ought to be in the skyline (false negatives). Identifying false negatives and reintroducing them into the skyline would not only improve the recall of the skyline, but also the precision, since the introduced false negatives would in turn dominate the false positives. Therefore, the advanced heuristic focuses on identifying false negatives for reintroducing them into the skyline.
- Crowd-based approach: This approach is based on *crowd-enabled databases* for completing values of incomplete tuples during runtime. Instead of naively crowdsourcing all missing data, we aim at selective hybrid approaches, which rely on crowdsourcing only those tuples where heuristics will most likely introduce errors, but fall back to using (cheap) heuristics when their application is safe. We present two different strategies as studied in previous works a) surrogating missing data with predicted values [2], b) surrogating missing data with min-max values [3].

This paper mainly summarizes and unifies previous work discussed in [2] and [3], but also provides new insights especially with respect to processing skylines heuristically. The rest of this paper is organized as follows: in the next section we discuss related work and give an overview on the theoretical foundations of skyline queries. In section 3 we introduce *common heuristics* for handling incomplete data and their effect on the result's quality. This is followed in section 4 with a detailed and extensive description of the two proposed approaches. Experiments are carried out in section 5, and we finally conclude with a summary.

2. Foundations and Related Work

In the following, we present the theoretical foundations of skyline queries on incomplete datasets. Furthermore, we introduce how the quality of a skyline result set for incomplete datasets can be measured, which allows us to evaluate both commonly used heuristics and our proposed approaches. For the crowd-based approach, we give an overview on crowd-enabled databases.

2.1. Crowdsourcing

Crowdsourcing has become a popular approach to many problems that cannot be easily addressed by automated methods and algorithms, or problems that explicitly require significant amount of human intelligence or human feedback.

In general, the term crowdsourcing may be attributed to any (Web) platform that explicitly or implicitly enlists a vast number of humans to collaboratively solve complex problems [Do11]. This ranges from explicit human collaboration efforts creating complex artifacts (e.g.,

Wikipedia or open source software) across sites based on user-generated content (e.g., YouTube) to sites implicitly exploiting human efforts by aggregating user opinions such as ratings or reviews (e.g., Netflix, IMDb, or Amazon.com), which can be exploited by for example recommender systems which utilize the input provided by the crowd [Be07]. Thus, the Social Web is also based on an extensive but mostly uncontrolled crowdsourcing effort. We will use a much narrower definition of crowdsourcing in the course of this paper: We will only focus on explicit crowdsourcing for general tasks based on controlled task execution as provided by Web services such as Amazon's Mechanical Turk, CrowdFlower, or ClickWorker, bringing crowdsourcing more in line with alternative concepts of 'work'. These Web platforms allow a complex task to be executed by dividing it into many smaller and simpler sub-tasks, i.e., HITs (Human Intelligence Tasks) – the smallest unit of crowdsourceable work, which are distributed to a human worker pool. In these platforms, workers are recruited and retained with payment, and the platform itself acts as a marketplace for different tasks and jobs. Hence, in theory such platforms can be used to perform any divisible tasks that require human intelligence.

These platforms have successfully been used by researchers from many different domains and knowledge processing tasks, e.g., disaster response [4], providing training data for machine-learning-based approaches [5], or performing large scale user studies for evaluating new prototype implementations [6], or performing surveys with a large and diverse number of participants for investigating general human behavior or preferences [7].

One of the main issues related to crowdsourcing is output quality [8][9][10]. Incentive mechanisms are also tightly linked to quality [11]. Different scenarios may require different incentive mechanisms. For instance, an industrial environment may require to provide economic incentives to motivate the crowd to perform certain tasks. The work presented in [12] presents some results that confirm the importance of money compared to other motivations in certain domains. The crowdsourcing scenarios discussed in this paper, i.e. completing of incomplete data records, is luckily one of the easiest scenarios with respect to quality management. As our scenario is only concerned with factual data that can be looked up on the Web without requiring expert knowledge (e.g., product specifications, telephone numbers, addresses, etc.), effective and simple quality control techniques like majority voting or Gold sampling [13] can be applied. In previous studies on crowdsourcing 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. For example, [13] reports that crowdsourced 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). Therefore, while quality issues are a severe concern for crowdsourcing in general, in this paper we simply assume that established quality control techniques are sufficient.

2.2. Skyline Queries and Incomplete Data

Skyline Queries [14] are a popular personalization technique for databases, which successfully bridge set-based SQL queries and top-k style ranking queries [15]. They implement the concept of *Pareto optimality* from economics, thus allowing for intuitive and simple personalization. Simply put, users provide for each relevant attribute a preference order, while assuming ceteris-paribus semantics (e.g., "lower prices are preferred to higher prices given that all other attributes are equal"). This implies, that 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).

The skyline set is computed by retrieving all tuples that are not dominated by any other tuple, i.e. all Pareto optimal tuples. In the basic case, where the underlying datasets are complete without missing values, skylines are defined as follows:

Definition 1 (Numerical Preferences and Pareto Dominance): *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 $a =_i b$.*

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 $t_1 \gtrsim_i t_2$ with respect to all attributes, and t_1 dominates t_2 with respect to at least one attribute:

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

A skyline query is given by a set of preferences $P = \{P_1, \ldots, P_n\}$, with one preference for each attribute. Accordingly, on the complete dataset R and a set of preferences P, the skyline sky is defined as:

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

On the other hand, when a dataset is incomplete, as is often the case in Web sources, the test for Pareto dominance cannot be directly performed, and no skyline can be computed without relying on additional heuristics. An incomplete dataset can be defined as follows, where missing values are denoted by □.

Definition 2 (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.*

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

2.1. Measuring Skyline Errors

For measuring the error introduced by a heuristic used in the case of missing information, we compare the skyline set obtained by heuristic handling to the set that would have been obtained had all information been available (i.e. compared to the skyline of the complete dataset). Basically, each heuristically handled tuple can be one of four cases: *true positive* (tuple is correctly included in the skyline), *true negative* (tuple is correctly excluded from the skyline), *false positives* (tuple is included in the skyline, but if all values had been available, it would have been excluded. This can result from the heuristic over-estimating the missing values, or from under-estimating another tuple, which under complete information would not have been dominated, but now fails to do so due to heuristic handling), *false negatives* (analogous to false positives, an incorrectly excluded tuple resulting from under-estimating this tuple or over-estimating another one).

For quantifying the total introduced error, we expand on the measure of informedness [16]. 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 or precision, it considers both error types: false positives and false negatives. At the same time, it also respects true positives and true negatives, which promotes the measure as quite fair and unbiased.

Definition 3 (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):*

$$error(sky_H, sky_R) = 1 - informdness(..)$$

$$inform.(sky_H, sky_R) = recall(..) + invRecall(..) - 1$$

$$recall(sky_H, sky_R) = \frac{\#truePositives(..)}{\#truePositives(..) + \#falseNegatives(..)}$$

$$invRecall(sky_H, sky_R) = \frac{\#trueNegatives(..)}{\#trueNegatives(..) + \#falsePositives(..)}$$

3. Simple Skyline on Incomplete data

In this section we take a close look at commonly used heuristics that have been so far utilized to support skyline computation on incomplete datasets. This allows us to select a suitable base heuristic for further improvement. For self-containment, we present the evaluation as carried out in previous work [3].

3.1. Basic Heuristics

For handling missing information in skyline computation, several commonly used basic heuristics are at hand, which provide default handling for the 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 on any potential good candidate.

We evaluated these heuristics on the same datasets which we used for all other experiments in this paper. We will abstain from experimenting with synthetic data, and evaluate instead using three real-world datasets from the area of sports and e-commerce for judging the performance of heuristics under realistic circumstances. Like most real life datasets, our practical datasets also show a higher degree of correlation. All our datasets were originally complete, and their values were artificially removed for the experiments. The datasets are:

1. The *NBA dataset* contains the NBA player statistics and has been frequently used within skyline research. The dataset comprises 21,961 tuples. For each player, only the following 5 attributes were retained: games played, points scored, rebounds, assists, and goals. With maximum preferences used for all attributes, a skyline of 75 tuples is yielded.
2. A *notebooks dataset* whose data was crawled from Dooyoo.de back in 2010. The dataset is made up of a total of 1,697 tuples, consisting of 6 attributes: CPU frequency, CPU type (categorical preference encoded by a score), RAM, HD, display size, weight (minimum preference). Unless stated otherwise, maximum preferences were used, yielding a skyline of 35 tuples.
3. A *cars dataset* holding information about different car models. The data crawled in 2011 from Heise.de spans over 7,755 tuples of 6 attributes; price, power (maximum preference), acceleration (maximum preference), fuel consumption, CO_2 emission and taxes. Minimum preferences were used unless explicitly stated otherwise, resulting in a skyline of 268 tuples.

Using these datasets, we covered the following heuristics (more detail about her result quality can be found in [3]):

Treat 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 and false positives. In our evaluation, this heuristic shows errors rates which are consistently worse than the pessimistic surrogation heuristic, but are considerable better than ISkyline or optimistic surrogation.

Treat incompleteness as incomparable: This conservative optimistic heuristic aims at minimizing false negatives, i.e. no tuple should be excluded from the skyline unless it can be clearly shown that it is dominated by another tuple. As the test for dominance cannot be performed when an incomplete tuple is involved, incomplete tuples don not dominate any other tuples, but also cannot be dominated themselves. Therefore, incomplete tuples will be in the skyline result, 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. The evaluation results of this heuristic in terms of error rates is comparable to treating incomplete tuples as a failure.

Surrogate with maximal values (optimistic surrogation): This optimistic heuristic differs from the previous two heuristics. Instead of providing a default decision for the dominance test, it heuristically provides the values of missing attributes, i.e. it simply surrogates every missing value with the best possible value. Then, the usual Pareto dominance semantics are applied for computing the skyline. The rationale behind this heuristic is that missing values might mask very good values, and in the best case, they might even be maximal. By assuming maximal values, this heuristic tries to retain these good tuples, yet at the same time leading to both (potentially many) false positives and false negatives. This also clearly shows in our evaluations where this heuristics shows error rates which are often 6x higher for all datasets than the best heuristic.

Surrogate with minimal values (pessimistic surrogation): This heuristic is similar to the previous heuristic, but takes a pessimistic approach surrogating missing values with the minimal value. This allows incomplete tuples to be in the skyline, 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 because their missing attribute values were assumed to be the minimal ones). In our evaluations, this heuristic was always producing significantly better results with respect to skyline quality, overshadowing the others by far. However, 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 crowd-sourcing.

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. In the following survey in subsection 3.2, k-nearest neighbor value (KNN) imputation will be used as it has been shown to be quite robust when the percentage of the missing values increases [17]. However, this heuristic works only well when the dataset is highly predictable. In consequence, we could observe very good results for the NBA dataset, but only mediocre error-rates for the cars and notebook dataset.

ISkyline Semantics: Among the recent studies, [18] defines a different skyline semantic that essentially alters the Pareto dominance definition to take missing values into consideration. This however, forsakes the skyline transitivity property and leads to cyclic dominance behavior. Accordingly, an alternative skyline computation algorithm ISkyline is introduced.

Unfortunately, from a quality perspective, this heuristic fares surprisingly badly. Its performance is comparable to optimistic surrogation.

4. High-Quality Skylines on Incomplete Data

In this section, we present two approaches for obtaining high-quality skylines on incomplete datasets: 1) an advanced heuristic approach, which aims at identifying and reintroducing false negatives for improved quality, 2) a crowdsourcing-based approach, which employs the help of human workers to provide missing data during runtime. Accordingly, a comprehensive comparison can be carried out later on in the evaluation section.

4.1. Advanced Heuristic Approach

The low quality of skyline sets computed over incomplete data can be attributed to one of two errors in handling incomplete tuples: a) the unjustified inclusion of tuples (false positives) and b) the omission of tuples that should be there (false negatives). False positives are present because they are not dominated by any other tuple i.e. tuples which should dominate the false positive are incomplete and the heuristic handling the test for Pareto domination missed this dominance. Such incomplete tuples end up being false negatives. Accordingly, the aim is to reintroduce these false negatives in the skyline, which would in turn dominate the false positives and discard them.

Relying on the previous study results in subsection 3.2, missing values are surrogated as per the *Minimal value surrogation* and the skyline is computed. Since the focus is on finding and reintroducing false negatives, the advanced heuristic operates only on the set of dominated tuples. Tuples identified as skyline tuples are ignored, since they can only be false (or true) positives.

To pinpoint these false negatives, the advanced heuristic operates more or less as a filter, getting rids of tuples deemed as true negatives. By discarding the true negatives, the remaining set of tuples then constitute the set of potential false negatives. The task of deciding whether a tuple is a false negative or a true negative becomes the main problem and can be based on three given guidelines. The guidelines can be thought off as consecutive sub filters, which are to be followed in that respective order.

First Guideline – exploits the implicit effect of the minimal value surrogation. Namely, an incomplete tuple´s real missing value will never be worse than the minimum value it is surrogated with. In other words, minimally surrogated skyline tuples will be more often than not true positives. In contrast, complete skyline tuples remain as potential false positives. The ability to distinguish between those two classes within the skyline yields two domination scenarios for the set of dominated tuples and more specifically the complete dominated tuples which could be either 1) dominated by a true positive, which implies that it is indeed a true negative, or 2) dominated by a false positive, which implies that there exists a tuple (currently a false negative) that should dominate this false positive and transitively then dominate the complete tuple in question. Accordingly, completely dominated tuples are always true negatives and can be filtered out. As will be demonstrated in the evaluation section, 80% of the dominated tuples are on average discarded, with only 5% of that mass being false negatives (i.e. 95% of the false negatives are preserved).

Second Guideline – enforces two bounds: an upper and a lower bound on the so-called *minimum skyline membership value.* A Minimum skyline membership value is simply the smallest value

assigned to a tuple's missing attribute upon which this tuple becomes a skyline point. Investigating whether an incomplete tuple has a minimum skyline membership value or not allows us to easily discard all those tuples who do not have this property, i.e. regardless of the missing attribute value, they´ll never be in the skyline. Instead of surrogating with the minimum value, we start to search for the minimum skyline membership value for each incomplete tuple in the set of dominated (non-skyline) tuples. Finding this minimum value involves an iterative approach. Starting off with the worst possible value *m*, the value incrementally increases by $m+x$ (where x is some fixed value referred to as the minimum skyline membership granularity) until the tuple under inspection qualifies into the skyline. A valid minimum skyline membership value must abide to the following lower and upper bounds.

- *Lower bound:* A potential false negative´s minimum skyline membership value shouldn´t qualify the tuple into the skyline by dominating a true positive (an incomplete tuple), but rather only false positives (complete tuples).
- *Upper bound:* A potential false negative's minimum skyline membership value shouldn't exceed the sum of the corresponding missing attribute's mean and minimum skyline membership granularity that is used.

Third Guideline - aims at narrowing in on the semantically important false negatives. We define semantically important tuples as those tuples having good overall attribute values, rather than one (as often suffices for Pareto dominance). To identify those semantically important tuples, we exploit the notion of subspaces skyline [21],[22]. A subspace skyline is a skyline that is computed on a subset of the dataset´s attributes. Examining the different subspaces aligns with definition of semantic importance, where some tuples may appear as skyline points in one or more subspace or in none. The focus here is on those potential false negatives who specifically emerge as *non-exclusive* subspace skyline points, i.e. a tuples that does not exclusively appear as a subspace skyline point in subspaces comprising its originally missing attributes. Such exclusivity would imply that the tuple's skyline-membership is solely based on the minimum skyline membership value that´s been surrogated with, and that none of the tuple's other attributes were good enough to qualify the false negative into the skyline set of any other subspace. On the other hand, non-exclusivity (i.e. appearing as skyline point in other different subspaces not comprising the originally missing attribute´s value) would imply that the tuple's overall attributes´ value are good.

4.2. Crowdsourcing-based Approaches

Crowd-enabled algorithms can be exploited to obtain missing values on the Web. Basically, HITs of incomplete tuples are issued during runtime to crowdsourcing services. While each individual HIT might be cheap, monetary cost and time needed to complete each HIT can quickly sum up. Accordingly, the naïve approach of crowdsourcing all missing attribute values of an incomplete dataset is prohibitively expensive, specially that most of the tuples completed by the crowd will not be part of the final result set, i.e. skyline set.

The next two approaches, which were investigated in previous works, balance these costs against the desired improvements by selectively crowd-sourcing the most relevant incomplete tuples, while relying on heuristics for all others.

Approach A - Surrogating with Predicted Values

This approach, first introduced in [2], surrogates missing values by (potentially unreliable) predictions before a skyline is computed. Tuples that are considered high impact (risk) have

their predicted values superseded with real values via crowdsourcing. Eventually, the skyline can be computed with high accuracy.

For predicting missing values of tuples, a library containing several predication algorithms is available. Different algorithms (e.g. k-nearest neighbour imputation [23], regression imputation [24], etc.) will perform differently depending on the properties of the underlying dataset. In a system tuning phase during start-up, the best prediction algorithms for the current dataset has to be determined. A test sample $R^S \subseteq R^C$ (whereas R^C is the subset of complete tuples) is prepared by removing some values in a similar pattern in which information is missing in the base dataset. Then, a prediction quality assessment is computed for each algorithm by measuring the error between original values in this test sample and the current algorithm's predictions for same values. The algorithm showing the smallest overall mean squared error is selected for predicting all missing values.

During evaluation we only consider k-nearest neighbour imputation (KNN) prediction algorithm, as it has been shown to be quite robust when the percentage of missing values increases (see [17], which also evaluates other imputation techniques).

Additionally, more detailed statistics on the behavior of each algorithm is stored to be used later. In particular, for each attribute a_i we elicit the mean error $\bar{e}_i$ between the predictions and the real values, and the standard deviation σ_i^e of those errors.

Definition 4 (Error Vector): *For prediction algorithm PA and base dataset R, an error analysis vector E_R^{PA} with e^{-2} as mean squared error of all attributes, $\bar{e}_i$ as mean error of attributea_i, and σ_i^e as respective standard deviation is given by:*

$$E_R^{PA} = (e^{-2}, \bar{e}_1, \dots, \bar{e}_n, \sigma_1^e, \dots, \sigma_n^e)$$

After the selected prediction algorithm has been used to predict all missing values for the current dataset, the system has to decide for each tuple, whether the tuple's missing values will be replaced by the predicted values during skyline processing or by accurate values that are obtained from crowdsourcing judgements. When surrogating with predicted values, a certain error is introduced. To control and minimize that error, we introduce the notion of *prediction risk*, i.e. the risk quantified by the probable impact each tuple may have on the skyline correctness whenever a predicted value is used without crowdsourcing the correct values. By computing this risk factor, we can restrict the crowdsourcing efforts to exactly those tuples that will strongly affect the result quality with high probability (i.e. have high risk), and use suitable value predictions for all remaining tuples posing only a limited threat to the overall result quality. Eventually, we only crowd-source the most *risky* tuples and rely on predicted values for all the other *safe* tuples.

Therefore, the next crucial step is to assess and classify the risk of each predicted tuple as being "risky" or "safe". To that end, we rely on the corresponding prediction algorithm accuracy statistics E_R^{PA} that were earlier elicited. With these statistics, we quantify the prediction interval of t^p assuming that the algorithm estimated any missing value including the respective systematic prediction error $\bar{e}_i$ and additionally overestimated or underestimated each value by the standard deviation, i.e. $\pm\sigma_i^e$. Then, the two interesting cases become the upper bound tuple t^+ and the lower bound tuple t^-, because these two tuples dominate the largest / lowest number of other tuples when finally computing the skyline under normal error assumption.

Definition 5 (Upper/Lower Bound Tuple): Let $t \in R^I$ be a tuple with incomplete values, and t^p be the predicted tuple using some prediction algorithm. Then the upper/lower bound tuples t^+ and t^- are defined attribute-wise as follows:

$$t_i^+ = \begin{cases} if\ (t_i \neq \square) : t_i \\ if\ (t_i = \square) : (t_i^p + \bar{e}_i) + \sigma_i^e \end{cases}$$

$$t_i^- = \begin{cases} if\ (t_i \neq \square) : t_i \\ if\ (t_i = \square) : (t_i^p + \bar{e}_i) - \sigma_i^e \end{cases}$$

Focusing on the expected errors for each predicted tuple t^p when computing the skyline of $R^C \cup \{t^p\}$, while assuming that the real values for t^p are bounded by t^+ and t^-, we identify for any given tuple the following four scenarios for all the possible false negatives and false positives.

Definition 6 (Set of False Positives): *Let $t^p = PA(t)$ be a predicted tuple with its upper/lower bound tuples t^+ and t^-. Also, sky_c is the skyline of all tuples in R^C with respect to the preferences P. Then, the set of possible false positives $fp(t^p)$ can be computed by the one of these four rules:*

- *If* $\left(\nexists\, s \in sky_c : (s >_P t^p)\right) \wedge \left(\nexists\, s \in sky_c : (s >_P t^-)\right)$ *then* $fp(t^p) = \left\{s \in sky_c \middle| (t^+ >_P s) \wedge \left(t^p \not\succ_v s\right)\right\}$

- *If* $\left(\nexists\, s \in sky_c : (s >_P t^p)\right) \wedge \left(\exists\, s \in sky_c : (s >_P t^-)\right)$ *then* $fp(t^p) = \left\{s \in sky_c \middle| (t^+ >_P s) \wedge \left(t^p \not\succ_v s\right)\right\} \cup \{t^p\}$

- *If* $\left(\exists\, s \in sky_c : (s >_P t^p)\right) \wedge \left(\exists\, s \in sky_c : (t^+ >_P s)\right)$ *then* $fp(t^p) = \{s \in sky_c | (t^+ >_P s)\}$

- *If* $\left(\exists\, s \in sky_c : (s >_P t^p)\right) \wedge \left(\nexists\, s \in sky_c : (t^+ >_P s)\right)$ *then* $fp(t^p) = \emptyset$

Definition 7 (Set of False Negatives): *Let t^p be a predicted tuple with its upper/lower bound tuples t^+ and t^-. Let sky_c be the skyline of all tuples in R^C with respect to the preferences P. Then, the set of possible false negatives $fn(t^p)$ can be computed by one of these four rules:*

- *If* $\left(\nexists s \in sky_c : (s >_P t^p)\right) \wedge \left(\forall\, s \in sky_c : (t^p >_P s) \Rightarrow (t^- >_P s)\right)$ *then* $fn(t^p) = \emptyset$

- *If* $\left(\nexists s \in sky_c : (s >_P t^p)\right) \wedge \left(\exists\, s \in sky_c : (t^p >_P s) \nRightarrow (t^- >_P s)\right)$ *then* $fn(t^p) = \left\{s \in sky_c \middle| (t^p >_P s) \wedge (t^- \not\succ_v s)\right\}$

- *If* $\left(\exists s \in sky_c : (s >_P t^p)\right) \wedge \left(\nexists s \in sky_c : (s >_P t^+)\right)$ *then* $fn(t^p) = \{t^p\}$

- *If* $\left(\exists s \in sky_c : (s >_P t^p)\right) \wedge \left(\exists s \in sky_c : (s >_P t^+)\right)$ *then* $fn(t^p) = \emptyset$

Using the cardinalities of the sets of false negatives $fn(t^p)$ and false positives $fp(t^p)$ for each predicted tuple t^p, we can finally assign a score that reflects the potential severity of the introduced error and accordingly rank the tuples with respect to their crowdsourcing priority.

Definition 8 (Tuple Score): *Given the set of false negatives $fn(t^p)$, false positives $fp(t^p)$, and a weighting factor $\alpha \in [0,1]$, the score of a tuple t^p can be computed as:*

$$score(t^p) = \alpha * |fn(t^p)| + (1 - \alpha) * |fp(t^p)|$$

Finally, the top-k most risky tuples can be crowdsourced. Different strategies for selecting a suitable k are detailed in [2], where k could be chosen in accordance to quality requirements or additional meta-data provided by the user.

Approach B – Surrogating with Min-Max Values

In this second crowdsourcing based approach [3], we rely on the previous study results in subsection 3.2, and surrogate all missing values with minimal values for a simple, but still

strong starting situation. As illustrated by the first guideline of the advanced heuristic, every tuple in the skyline's result set that has been surrogated with minimal values has now a very high probability to be a true positive and shouldn't be crowdsourced.

Next we introduce an error model for identifying only those tuples with the highest potential to negatively affect quality, and select only those to be crowdsourced. The error model relies on counting the number of tuples dominated by a given tuple when its missing values have been surrogated. Though minimal values surrogation proves to be the best baseline heuristic and has its advantages, naïvely surrogating the tuples values with the minimum value becomes less effective underneath this error model's setting, where only few or even no other tuples are usually dominated when surrogating with minimal values. Furthermore, it would also ignore the possible potential of tuples, as usually most real values are better than the worst value. Accordingly, we temporarily surrogate the current tuple's missing values instead with maximum values, while retaining the minimal values surrogation for all the other incomplete tuples to be ranked with minimal values. 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 crowdsourced. Every time a tuple is crowdsourced, this ranking is recomputed to adapt to the changes of the new information and reflect it upon the skyline's result.

Definition 9 (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:*

$$maxDomCount(t) = |\{t_d \in R^C | \hat{t} >_P t_d\}| \text{ with}$$
$$\hat{t}_i = \begin{cases} 1 : if\ t_i = \square \\ t_i : if\ t_i \neq \square \end{cases}$$

Furthermore, we incorporate the notion of missing attributes' skyline impact, arguing that not all attributes have an equal impact on the skyline's result, and that some attributes can be more influential for deciding a tuple's membership in the skyline than others. In this error model, the potential impact of all attributes is measured in an initial dataset analysis step. 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, 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 error model for an improved ranking. Eventually, tuples are ranked by their weighted minimal-maximal domination count as follows:

Definition 10 (Missing Attribute's Skyline 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:

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

TABLE VI INITIAL DATA ANALYSIS (CARS DATASET)

Cars dataset	***KNN*** e^{-2}: *0.077*		***IV***
	$\lvert\bar{e}_i\rvert$	σ_i^e	
Price	0.25	0.5	78.534
Power	0.10	0.12	90.312
Acceleration	0.22	0.34	33.609
Fuel Consumption	0.20	0.35	28.758
CO^2 Emission	0.21	0.35	10.848
Taxes	0.26	0.4	47.775

TABLE VII INITIAL DATA ANALYSIS (NBA DATASET)

NBA dataset	*KNN* e^{-2}: *0.0510*		***IV***
	$\lvert\bar{e}_i\rvert$	σ_i^e	
Games played	0.34	0.38	62.667
Points scored	0.08	0.08	2.667
Total rebounds	0.05	0.07	50.667
Assists	0.08	0.07	78.667
Field goals made	0.06	0.07	6.667

TABLE VIII INITIAL DATA ANALYSIS (NOTEBOOKS DATASET)

Notebooks dataset	*KNN* e^{-2}: *0.093*		***IV***
	$\lvert\bar{e}_i\rvert$	σ_i^e	
CPU	0.36	0.39	8.632
CPU Frequency	0.25	0.35	40.181
RAM	0.25	0.31	17.323
Hard drive	0.15	0.21	25.835
Display Type	0.24	0.40	68.571
Weight	0.15	0.35	88.571

TABLE IX INITIAL DATA ANALYSIS (CARS DATASET)

Cars dataset	**KNN** e^{-2}: 0.077		***IV***
	$\lvert\bar{e}_i\rvert$	σ_i^e	
Price	*0.25*	*0.5*	*78.534*
Power	*0.10*	*0.12*	*90.312*
Acceleration	*0.22*	*0.34*	*33.609*
Fuel Consumption	*0.20*	*0.35*	*28.758*
CO^2 Emission	*0.21*	*0.35*	*10.848*
Taxes	*0.26*	*0.4*	*47.775*

TABLE X INITIAL DATA ANALYSIS (NBA DATASET)

NBA dataset	KNN e^{-2}: 0.0510		***IV***
	$\lvert\bar{e}_i\rvert$	σ_i^e	
Games played	*0.34*	*0.38*	*62.667*
Points scored	*0.08*	*0.08*	*2.667*
Total rebounds	*0.05*	*0.07*	*50.667*
Assists	*0.08*	*0.07*	*78.667*
Field goals made	*0.06*	*0.07*	*6.667*

TABLE I INITIAL DATA ANALYSIS (NOTEBOOKS DATASET)

Notebooks dataset	KNN e^{-2}: 0.093		***IV***
	$\lvert\bar{e}_i\rvert$	σ_i^e	
CPU	*0.36*	*0.39*	*8.632*
CPU Frequency	*0.25*	*0.35*	*40.181*
RAM	*0.25*	*0.31*	*17.323*
Hard drive	*0.15*	*0.21*	*25.835*
Display Type	*0.24*	*0.40*	*68.571*
Weight	*0.15*	*0.35*	*88.571*

TABLE II AVERAGE % OF FALSE NEGATIBES RETAINED

Guidelines \ **Data**	Cars	NBA	Notebooks
First Guideline	*95.679%*	*100%*	*84.751%*
Second Guideline - Lower bound -	*93.349%*	*100%*	*82.901%*
Second Guideline - Upper bound-	*47.4%*	*5.859%*	*32.667%*
Third Guideline	*35.435%*	*4.164%*	*25.383%*

TABLE III AVERAGE % OF DISCARDED DATA

Guidelines \ **Data**	Cars	NBA	Notebooks
First Guideline	*79.653%*	*79.97%*	*79.814%*
Second Guideline - Lower bound -	*84.042%*	*91.202%*	*85.487%*
Second Guideline - Upper bound-	*97.212%*	*99.99%*	*98.89%*
Third Guideline	*98.948%*	*99.99%*	*99.558%*

TABLE IV ADVANCED HEURISTIC'S SKYLINE QUALITY

ID	*CPU*	*CPU Freq.*	*RAM*	*HD*	*Display Type*	*Weight*
1415	*0.8*	*0.751*	*0.5*	*0.4*	*0.846*	*0.628*
156	*0.502*	*0.751*	*0.5*	*0.5*	*0.846*	*0.667*
1292	*0.8*	*0.863*	*1*	*0.64*	*0.666*	*0.79*
1332	*0.8*	*0.737*	*0.5*	*0.32*	*0.851*	*0.603*

TABLE V ADV. HEURISTIC'S SKYLINE SIZE

Datasets	Cars dataset	NBA dataset	Notebooks dataset
sky_A	*380*	*80*	*48*
$sky_{p^{min}}$	*296*	*75*	*35*

5. Evaluation

In this section, we evaluate both advanced heuristic and crowdsourcing based approaches and investigate which approach produces a higher quality result. Unless stated otherwise, we assume that 20% of the values are missing. For efficiency reasons, all attribute values are normalized to the interval [0, 1], with 1 being the best and 0 being the least desirable value. Furthermore, seeking to avoid dataset bias, all experiments were executed for a total of 100 simulation runs. The presented results are an aggregation of these runs. For the crowdsourcing based approach an initial dataset analysis step is required to measure both the KNN's prediction quali ty (for approach A: Surrogating with predicted values) and the missing attributes' skyline impact (for approach B: Surrogating with Min-Max values). These measurements are necessary prerequisites, which give insights into the nature and predictability of the datasets.

5.1. Initial Dataset Analysis Step

Each attribute´s skyline impact values *IV* are obtained by measuring the introduced skyline's error, when the corresponding attribute is ignored or completely missing (in Table I, II and III). Some attributes instantly stand out, displaying a more influential role than others. In the Notebooks dataset, *Weight*, *Display Type* and *CPU Frequency* have more sway on a tuple being in the skyline than the *CPU*. Similarly, in the Cars dataset the *Power* and *Price* should be carefully considered when missing. A tuple missing the value for the *Power* attribute should be treated as more error-prone than a tuple missing the *$C0^2$-Emission*.

Furthermore, the prediction error of the KNN's algorithm, given by the error vector E_R^{KNN} for all the datasets is also shown in Table I, II, and III. It becomes clear that on average, missing values within the NBA dataset can be predicted significantly better than those of the other datasets. Moreover, even within a dataset, some attributes can be predicted more accurately than others, e.g., games played in the NBA set has a mean error of ~0.34, while most other attributes of the NBA set have mean errors of ~0.07. Evaluations with other prediction algorithms can be found in [2].

5.2. Evaluating the Proposed Approaches

Advanced Heuristic Approach

Ideally, a high percentage of semantically important false negatives should be retained, while discarding as many true negatives as possible. To conduct a comprehensive evaluation, the data set produced after each of the three guidelines is analyzed by inspecting: 1) the percentage of false negatives retained (see Table IV), which was inferred by comparing the data returned after following each guideline against the original skyline on the complete dataset and 2) the percentage of tuples discarded (see Table V), which was inferred by comparing the data returned after following each guideline against the initial set of dominated tuples (i.e. non-skyline tuples) from the minimally surrogated dataset that was used as input.

On average, 82 tuples remain for the cars dataset, 2 tuples for the NBA dataset and 7 tuples for the Notebooks dataset. Simultaneously, as depicted in Table IV, 35% for the cars dataset (~18 false negatives) and 25% for the notebooks dataset false negatives are retained.

Furthermore, we closely examine the semantic importance of the retained tuples. Using the original complete dataset and its skyline, we can check which tuples from the set of real false negatives were kept and compare it with those that didn't. The earlier experiments already illustrated how many false negatives were retained, but the main aim here is to check whether

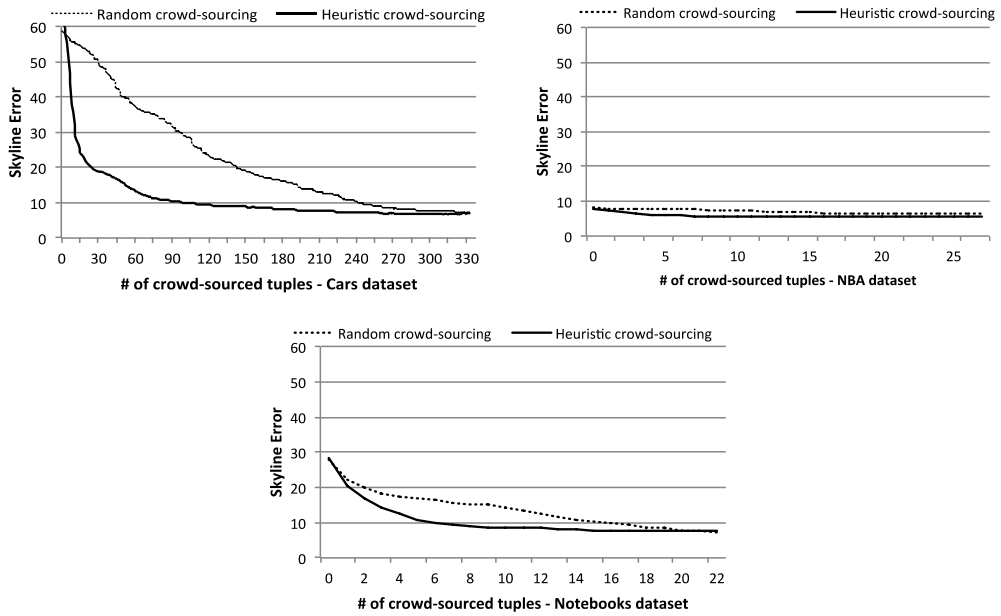

Figure. 1. Decreasing skyline error while crowdsourcing (heuristic and random tuple selection)

the false negatives that were kept are indeed more interesting than those discarded. Without loss of generality we show the results of a single run on the notebooks dataset. For the current simulation run, the computed skyline lacks a total of 4 false negatives (note that, the original skyline is also small in size, including only 35 tuples.) Table VI represents the attribute values of those four false negatives. Out of 1,658 tuples, the advanced heuristic approach returns only 7 data points, including two false negative, namely, data points with IDs: 1415 and 156 and misses out on: 1292 and 1332. Keeping in mind that the highest attributes with impact are in order as follows (as per the initial dataset analysis strep): Weight, Display type and CPU frequency; a closer look on the four false negatives arguably demonstrates that the two returned false negatives are the most interesting. Although the Notebook with ID 1292 has the highest CPU frequency, it still has the smallest display and is the heaviest. On the other hand, for Notebook ID 1332, though -compared to the retained notebooks 1415 and 156 – it's the lightest and have the biggest display, it's not included as the difference in terms of weight and display type isn't that big, while at the same time, it ranks equally or smaller in terms of all the other attributes.

Crowdsourcing based Approaches

To examine the effectiveness of the two proposed crowdsourcing based approaches, we measure the skyline error every time a tuple is crowdsourced.

Approach A - Surrogating with predicted values

Fig. 1 depicts the results of the selecting the next tuple to be crowdsourced, instead of randomly selecting any incomplete tuple for crowdsourcing. With the notable exception of the NBA dataset, it is clearly visible that by pure prediction (i.e. no crowdsourcing) the skyline quality over incomplete Web sources is very low (error for NBA 7.2%, notebook 28.4%, cars 62%). However, the skyline error is significantly reduced when using this heuristic after just a few crowdsourced tuples, while this effect is much less pronounced if tuples are randomly selected. For example, consider the cars dataset: for decreasing the error from 60% down to 20% only 27 tuples needs to be crowdsourced on average, while for reaching a similar improvement with

randomly selected tuples, 145 tuples need to be crowdsourced. Also note that 20% missing values translate to an absolute of 4,392 tuples missing in the NBA dataset, 319 tuples in the notebooks dataset, and 1,551 tuples in the cars dataset. Therefore, tremendous effort can be saved, if users are willing to accept minor reductions with respect to skyline quality. The high prediction accuracy of the NBA dataset leads to an already low initial skyline error when only relying on prediction, leaving only limited room for improvement. Furthermore, this experiment shows another interesting effect which can be exploited: when using our heuristic, the error is reduced very quickly for the first few crowdsourcing operations, but the quality improvements will slow down after a while. This means, for most datasets, there is a *k*, for which we have the optimal trade-off between low error rates, and low query execution costs. We can use this observation to automatically determine the most efficient *k* during sampling runs prior to the actual crowdsourcing. This can be attained by deter mining the error curve's inflection point. E.g. using the cars dataset, by crowdsourcing 36 or 76 tuples (both being inflection points), a very good ratio between quality and costs can be achieved.

Approach B – Surrogating with min-max values

As shown in Fig. 2, applying this model on the cars dataset, the skyline error decreases from 10.8 to 9.8 instead of to only 10.4 for just 25 crowdsourcing operations. This significantly outperforms the KNN-predicted values surrogation variant whose skyline error starts high at 61.8 and decreases to only 19.8 after 25 crowdsourcing operations.

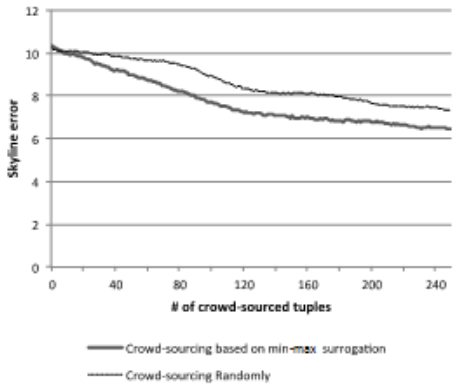

Figure 2. Crowdsourcing based on min-max value surrogation versus random crowdsourcing

5.3. Crowdsourcing based approaches Versus Advanced heuristic approach

In the next set of experiments, we compare the advanced heuristic approach to the second outperforming crowdsourcing approach: surrogating with min-max values in terms of: precision, recall, skyline error, and skyline size. We examine the skyline size to keep track of how big the skyline gets when we add the potential false negatives to it, where big skylines would overwhelm the user.

Skyline size

As shown in table VII, the skyline size of the advanced heuristic (sky_A), post min-max values surrogation baseline skyline ($sky_{D^{min}}$), are compared for the three datasets. As expected, sky_A is bigger than $sky_{D^{min}}$, since it reintroduces new tuples into the skyline. However, the difference in size isn't considerable for both the notebooks (13 tuples

TABLE XI ADV. HEURISTIC'S SKYLINE QUALITY

		sky_A	$sky_{D^{min}}$
Cars dataset	*Precision*	*0.658*	*0.818*
	Recall	*0.932*	*0.901*
	Skyline error	*8.5%*	*11.34%*
NBA datasets	*Precision*	*0.872*	*0.89*
	Recall	*0.93*	*0.927*
	Skyline error	*7.5%*	*7.59%*
Notebooks dataset	*Precision*	*0.7*	*0.791*
	Recall	*0.92*	*0.88*
	Skyline error	*9.06%*	*11.23%*

more) and the NBA dataset (5 tuples more). This increased size comes at the expense of sky_A having more correct skyline points than $sky_{D^{min}}$.

Precision, recall and skyline error

Table VIII holds the computational results for the precision, recall and skyline error. Notably sky_A's recall is higher than $sky_{D^{min}}$'s as it suffers from less false negatives in total. On the other hand, because sky_A's has more false positives and so exhibits a slightly smaller precision than that of $sky_{D^{min}}$'s. The skyline error based on the informedness measure attests that sky_A has smaller or roughly equal associated error.

Furthermore, compared to the crowdsourcing strategy, a real crowdsourcing experiment on the cars dataset required 109 crowdsourcing operations to reach the same skyline error achieved by the advanced heuristic strategy (8.5%).

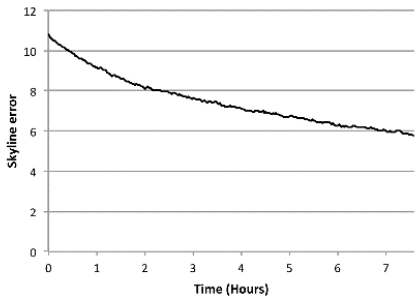

Figure 3. Time required for Crowdsourcing – Cars dataset

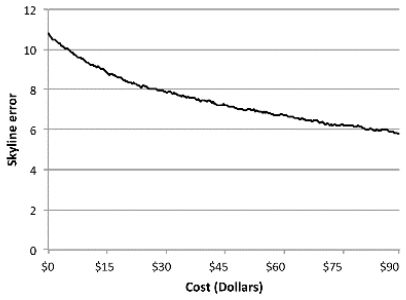

Figure 4. Cost for Crowdsourcing in dollars – Cars dataset

5.4. Crowdsourcing's costs and time

In a real crowdsourcing experiment that focuses on monetary and time costs, we evaluate the efficiency of the min-max value surrogation variant of the crowdsourcing based approach. We used CrowdFlower.com as a crowdsourcing platform, and again chose the cars dataset with 20% missing values to run our experiment. Starting off with a skyline based on the min-max surrogation heuristic, we crowdsourced one tuple at a time and then we measured the skyline error after each crowdsourcing operation. In order to obtain reliable values from the crowd-workers, for every crowdsourced value, a majority vote from 4 workers was required for quality control, i.e. each single value was crowdsourced 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. Fig. 3 and Fig. 4 illustrate the skyline result error improvement from 10.8% to 5%. In the end, crowdsourcing 250 out of the 7,755 overall tuples roughly required 7.5 hours and $89.

6. Conclusion

In this paper, we studied the problem of skyline query computations on incomplete Web datasets. Two strategies were proposed: Advanced heuristic and crowdsourcing-based. Whereas the crowdsourcing strategy exploits the current trend of crowd-enabled DBMS, which can attain strong results by incorporating human workers, the advanced heuristic offers an alternative offline solution for times when crowdsourcing might not be a feasible option, e.g. when the missing data is not easily available for the crowd or the costs of crowdsourcing are prohibitive. A survey of the most common established heuristics was conducted, showcasing how each of these heuristics fare in terms of the resulting skyline quality. We conducted all of our experiments on three originally complete real world Web datasets, where 20% of the data was later on artificially removed to induce incompleteness. This allowed us to examine the quality of heuristically computed skylines, as the error with regard to the corresponding complete dataset's skyline can be computed. Experiments illustrate how the advanced heuristic surpasses all the current basic heuristics that are employed in the literature. On the other hand

for the crowdsourcing based approach, min-max values surrogation is significantly better than KNN surrogation. Moreover, when comparing the advanced heuristic to the crowdsourcing approaches, it becomes clear that for some datasets, like NBA and Notebooks, the simple advanced heuristic outperforms the crowdsourcing alternative in terms of skyline quality though at the expense of an arguably insignificant skyline size growth.

7. 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. --- blinded ---
3. --- blinded ---
4. Goodchild, M., Glennon, J.A.: Crowdsourcing geographic information for disaster response: a research frontier. Int. J. Digit. Earth. 3, 231 (2010).
5. Selke, J., Lofi, C., Balke, W.-T.: Pushing the Boundaries of Crowd-Enabled Databases with Query-Driven Schema Expansion. Proc. VLDB. 5, 538–549 (2012).
6. Kittur, A., Chi, E.H., Suh, B.: Crowdsourcing user studies with Mechanical Turk. SIGCHI Conf. on Human factors in computing systems (2008).
7. Bentivogli, L., Federico, M., Moretti, G., Paul, M.: Getting expert quality from the crowd for machine translation evaluation. Proceedings of the MT Summmit. pp. 521–528 (2011).
8. Venetis, P., Garcia-Molina, H.: Quality control for comparison microtasks. Int. Workshop on Workshop on Crowdsourcing and Data Mining. , Bejing, China (2012).
9. Hirth, M., Hoβfeld, T., Tran-Gia, P.: Cost-Optimal Validation Mechanisms and Cheat-Detection for Crowdsourcing Platforms. Int. Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS). , Soul, South Korea (2011).
10. Kazai, G., Kamps, J., Milic-Frayling, N.: The face of quality in crowdsourcing relevance labels: demographics, personality and labeling accuracy. Int. Conf. on Information and Knowledge Management (CIKM). , Maui Hawaii, USA (2012).
11. Kamar, E., Horvitz, E.: Incentives for truthful reporting in crowdsourcing. Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS). , Valencia, Spain (2012).
12. Silberman, M.S., Irani, L., Ross, J.: Ethics and tactics of professional crowdwork. XRDS Crossroads. 17, 39–43 (2010).
13. Lofi, C., Selke, J., Balke, W.-T.: Information Extraction Meets Crowdsourcing: A Promising Couple. Datenbank-Spektrum. 12, (2012).
14. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. Int. Conf. on Data Engineering (ICDE). , Heidelberg, Germany (2001).
15. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. Symposium on Principles of Database Systems (PODS). , Santa-Barbara, California, USA (2001).
16. Powers, D.M.W.: Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Sch. Informatics Eng. Flinders Univ. Adelaide Aust. Tech Rep SIE07001. (2007).
17. Acu, E.: The treatment of missing values and its effect in the classifier accuracy. Classif. Clust. data Min. Appl. 1–9 (2004).
18. Khalefa, M.E., Mokbel, M.F., Levandoski, J.J.: Skyline Query Processing for Incomplete Data. Int. Conf. on Data Engineering (ICDE). , Cancún, México (2008).
19. 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).
20. Godfrey, P.: Skyline cardinality for relational processing. How many vectors are maximal? Symp. on Foundations of Information and Knowledge Systems (FoIKS). , Vienna, Austria (2004).
21. Jian Pei, Wen Jin, Martin Ester, Yufei Tao: Catching the best views of skyline: a semantic approach based on decisive subspaces. 31st Int. Conf. on Very Large Databases (VLDB'05). , Trondheim, Norway (2005).
22. Vlachou, A., Vazirgiannis, M.: Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships. Data Knowl. Eng. 69, 943–964 (2010).
23. Acu, E.: The treatment of missing values and its effect in the classifier accuracy. Classif. Clust. data Min. Appl. 1–9 (2004).
24. Ghahramani, Z., Jordan, M.I.: Supervised learning from incomplete data via an EM approach. In: Cowan, J.D., Tesauro, G., and Alspector, J. (eds.) Advances in Neural Information Processing Systems 6. pp. 120–127. Morgan Kaufmann Publishers, Inc. (1994).