The VLDB Journal (2024) 33:1283–1306
https://doi.org/10.1007/s00778-024-00849-w

REGULAR PAPER

Data distribution tailoring revisited: cost-efficient integration of representative data

Jiwon Chang[1] · Bohan Cui[1] · Fatemeh Nargesian[1] · Abolfazl Asudeh[2] · H. V. Jagadish[3]

Received: 15 July 2023 / Revised: 9 January 2024 / Accepted: 13 March 2024 / Published online: 12 April 2024

Abstract
Data scientists often develop data sets for analysis by drawing upon available data sources. A major challenge is ensuring that the data set used for analysis adequately represents relevant demographic groups or other variables. Whether data is obtained from an experiment or a data provider, a single data source may not meet the desired distribution requirements. Therefore, combining data from multiple sources is often necessary. The data distribution tailoring (DT) problem aims to cost-efficiently collect a unified data set from multiple sources. In this paper, we present major optimizations and generalizations to previous algorithms for this problem. In situations when group distributions are known in sources, we present a novel algorithm RatioColl that outperforms the existing algorithm, based on the coupon collector's problem. If distributions are unknown, we propose decaying exploration rate multi-armed-bandit algorithms that, unlike the existing algorithm used for unknown DT, does not require prior information. Through theoretical analysis and extensive experiments, we demonstrate the effectiveness of our proposed algorithms.

1 Introduction

The standard assumption in machine learning is that we have, at hand, a training data set that is a representative sample of the data that will be seen in production. This assumption is easily satisfied if the training data can be obtained by randomly sampling from the "full" data set in production. However, such random sampling is frequently not possible. Often, this is because production data has not yet been generated at the time the model is trained. At other times, the entire point may be to repurpose and reuse data collected for other purposes. Insufficiently representative training data has resulted in many data science debacles [2–5].

Even when the distribution is accurately characterized, it may not be so easy to obtain training data from the same distribution. For example, surveys may be sent out to a carefully chosen random sample, but only a fraction of the surveys are returned, with the return rate not being completely random (Table 1). Survey statistics has developed sophisticated techniques to handle such lack of randomness [6]. Similar issues arise when analyzing online comments or tweets to gauge popular opinion. We wish that the opinions expressed by representatives of the target population of interest (e.g., all voters or all customers), but we know that we only have a skewed sample with the most vocal individuals, potentially skewing young and more tech-savvy individuals. Beyond the need for representation to reduce model error, it may sometimes be important to show adequate consideration of minority groups. Even where representative samples can be obtained for training data, that still may not be sufficient in some circumstances. To ensure that minority entities are adequately considered, we may need to train with data in which small minorities are intentionally over-represented [7, 8]. Similarly, when we are interested in characteriz-

This research is supported in part by NSF 1741022, 2107290, 1934565, 2107050, the Google research scholar award, and the Schwartz Discover Grant.

✉ Jiwon Chang
jchang38@ur.rochester.edu

Bohan Cui
bcui2@u.rochester.edu

Fatemeh Nargesian
fnargesian@rochester.edu

Abolfazl Asudeh
asudeh@uic.edu

H. V. Jagadish
jag@umich.edu

[1] University of Rochester, Rochester, NY, USA

[2] University of Illinois Chicago, Chicago, Illinois, USA

[3] University of Michigan, Ann Arbor, Michigan, USA

Table 1 Comparison of algorithms

Scenario	Algorithm	Section	Space	Time (Initial)	Time (Per Iter.)	Bounds
Known	DP [1]	§ 3.1	$O(\prod_{i=1}^{m} Q_i)$	$O(nm \prod_{i=1}^{m} Q_i)$	$O(1)$	Optimal
Known	CoupColl [1]	§ 3.2	$O(n+m)$	$O(nm)$	$O(m)$	Theorem 1
Known	RatioColl	§ 3.3	$O(n+m)$	$O(nm)$	$O(m)$	Theorem 1, 2
Unknown	UCB [1]	§ 4.2	$O(n+m)$	$O(n+m)$	$O(n)$	$O(\log T)$ regret
Unknown	EpsilonGreedy	§ 4.3	$O(nm)$	$O(nm)$	$O(nm)$	Theorem 3
Unknown	ExploreExploit	§ 4.3.3	$O(nm)$	$O(nm)$	$O(m)$	None

Each algorithm has a straightforward duplicate-aware variant. We discuss the duplicate-aware variants of the algorithms in § 3.2.2. This modification requires $O(nm + Q)$ space for bookkeeping, $O(nm)$ initial time to set up trackers, and $O(nm)$ time per iteration to recompute heuristics

ing rare events, we may need training data that has rare events over-represented. For example, to learn how to handle emergencies, we need car-driving data with accidents and near-accidents over-represented: representative driving data may involve few challenging scenarios [9].

Data scientists may also need to acquire additional data to fix an existing model. Collecting data indiscriminately could introduce data imbalance issues and incur unnecessary costs. For instance, it is common for models to perform poorly on just a small subset of feature combinations, or slices. The problem of finding such slices efficiently has garnered attention in the data management community [10, 11]. Prior work also demonstrated how to compute the optimal amount of additional data per slice by estimating their learning rates and correlations [12]. Such methods may be used to obtain sophisticated distribution requirements. To summarize, data scientists often have distribution requirements on data sets they wish to use for training or analysis.

To see how to meet these requirements, we now turn to where the data come from. Sometimes, the data may explicitly be collected by the data scientist for the analysis at hand, using surveys, sensors, or other data collection means. Alternatively, data scientists could rely on secondary data instead: using data that have been collected previously for some other purpose. The number and variety of data sources available has been increasing rapidly, making secondary data analysis much more attractive. In fact, the data scientist on many occasions may be spoiled for choice. Since each data source is collected in some manner over some population, it will have its own distribution, which may differ from the distribution desired by the data scientist. The question to ask then is whether data from multiple sources can be mixed to achieve the desired distribution. This is the central problem we study in this paper.

Example 1 A data science company has been asked to build an ML model for a local bank in Texas who wants to offer a loan to employees with yearly income of more than $75K. The model should predict the likelihood that an individual will pay back the loan. The company considers building a model on an in-house data set. Being aware of recent incidents of racial/gender biases in similar predictive tools [13], the company wants to make sure different demographic groups are suitably considered. It, however, turns out the data set is skewed: while around 40% of samples are `white male`, it only 15% are `non-white female`. The company realizes there are alternative external data sources (such as *TexasTribune* [1]) they could consider for collecting the data. It establishes a target distribution on counts from different demographic groups (e.g., 25% from each demographic group in a data set of 1K samples). The challenge the company faces is how to efficiently query these data sources to collect the data.

Obtaining data from a data source is not free. Our focus lies on the most costly cases. An increasingly common situation where the costs are explicit is when data is purchased from a commercial data provider [14–18]. Data acquisition via survey sampling or crowdsourcing human annotations [19] also involves monetary costs for labor. Even for primary data collection there is a cost per tuple, in terms of access, storage, indexing, and so on. The access cost is especially high if data sources are complex join queries over large, relational data. In all cases, we can characterize the cost of obtaining data from any source in a pricing model. Given a set of these data sources, each with its own distribution and pricing model, our goal is to obtain, at least cost, an aggregate data set that satisfies our distribution requirements. This problem is difficult to solve in general because each source has its own distribution, and none may have a distribution that we seek. Furthermore, no combination of sources may provide us with the desired distribution either. In general, we may have to over-purchase and then "throw away" excess data items. And even so, we cannot be guaranteed it is feasible to obtain the desired distribution.

Summary of Results In our prior work, we proposed sampling strategies for two scenarios: (1) when the metadata about the data distribution is available and (2) when sources are opaque with unknown distributions [1]. For

[1] https://salaries.texastribune.org.

known distributions, we proposed a strategy based on the Coupon Collector's problem. For unknown distributions, we adopted a multi-arm bandit strategy with a customized reward function. In this manuscript, we revisit the proposed strategies and, building upon our observations, present alternative strategies and show their superiority. In particular, in the prior work, the (empirical) distribution of groups in sources and the costs of sources are considered the main factors in the source selection strategy. Intuitively, the strategies were designed to focus on collecting data for minorities at the lowest possible cost per unit, because minorities are less likely to be sampled from an arbitrary source than majorities. While this strategy is effective, our observation is that when a planner has reached the point that requires collecting a large number of majorities compared to minorities, prioritizing majorities can lead to a more cost-effective plan. In this manuscript, we present theoretical and empirical results suggesting that an adaptive sampling strategy with the consideration of the remaining group count requirements at different iterations is crucial to cost estimation and optimization. The following are our specific contributions.

- We reexamine the existing work on the *Data distribution tailoring* (DT) problem [1]. (§ 2)
- We propose an improved algorithm, called RatioColl, based on a heuristic that simultaneously prioritizes rare groups and groups with high remaining query counts. (§ 3)
- We generalize the problem to scenarios where distributions are unknown and no prior information about overall demographics are available. We propose an ε-greedy multi-armed bandit algorithm, called EpsilonGreedy, that uses the same objective function as RatioColl. (§ 4)
- We establish a tight asymptotic expected cost on RatioColl under constrained scenarios, and a sublinear asymptotic regret for EpsilonGreedy.
- To validate and evaluate the performance of the proposed algorithms, we conduct comprehensive experiments on real and synthetic data. (§ 5)

2 Problem definition

In this section, we formally define the data distribution tailoring (DT) problem. We adopt the framework proposed in [1], with some modifications to the notations and assumptions. The notations are listed in Table 2. Formally, an instance of the DT problem is represented by data sources $\mathcal{D}$, groups $\mathcal{G}$, query costs $\mathcal{C}$, and query requirements $\mathcal{Q}$. The output is a unified data set O. We elaborate on each element below.

Table 2 Table of notations

Symbol	Description
n	Number of data sources
D_i	a data source
G_j	a group
m	Number of groups
Q	Total query count, i.e., $Q = \sum_{j\in[m]} Q_j$
Q_j	Desired number of tuples from group G_j
C_i	Cost of sampling from D_i
N_i	Number of samples taken from D_i
$N_{i,j}$	Number of samples of G_j taken from D_i
$N'_{i,j}$	Number of unique samples of G_j taken from D_i
O	Collected target data set so far
G^*	Priority group at current iteration
D^*_j	Data source with minimum expected cost of collecting an item of G_j at current iteration
$P_{i,j}$	Proportion G_j in D_i, i.e., $\lvert G_j \cap D_i\rvert/\lvert D_i\rvert$
$P'_{i,j}$	Proportion of non-duplicate tuples of G_j in D_i, i.e., $\lvert G_j \cap D_i \setminus O\rvert/\lvert D_i\rvert$
P^j	Overall frequency of G_j in all data sources, i.e., $\lvert G_j\rvert/\lvert\bigcap_{i\in[n]} D_i\rvert$
	i.e., $\lvert G_j \setminus O\rvert/\lvert\bigcap_{i\in[n]} D_i\rvert$
t	Total number of samples taken so far

2.1 Data sources

The first input of the DT problem is a collection of sources $\mathcal{D} = \{D_1, \ldots, D_n\}$. We assume there is a way of unifying the schemata of sources and the user's target schema [20] and each tuple in a source can be associated with a group. This can be done by inspecting its sensitive attributes or annotating using classifiers and crowdsourcing. Data sources can be external, accessible through limited interfaces or APIs, or data views that are the outcome of the discovery and integration over underlying data sets.

In general, web services such as Google Flights API [21], open data lakes, such as `data.gov` and `CKAN API`, data markets such as Dawex [14], Xignite [15], and WorldQuant [16], as well as data brokers [17, 18] are examples of external data sources. Another setting is crowd-based data annotation and collection. Similar to incentive-based [22], distribution-aware [23], and cost-effective [24] crowdsourcing, each worker can be considered as a source providing annotated data with a possibly unknown distribution and potential bias. This calls for adaptive worker selection. Finally, when data sources are collections of tabular data sets, a source may be defined by a project-join query defined over a database or a data lake [25].

Sometimes obtaining a source with the same schema as the target schema requires data integration using a projection-join query over data sets that contain some attributes of the query. Continuing with the loan approval example, a data source using the TexasTribune database [26], the query $\Pi_{\texttt{race,gender,income,...}}(\texttt{employees} \bowtie \texttt{salary} \bowtie \texttt{loans})$ provides a data source. Of course, since the target schema is user-specific, and given the potentially large size of data sets, computing and materializing the full join for all sources is not efficient. Instead of offline join, existing work proposes ways for obtaining independent and/or uniformly distributed random tuples from the result of join without executing the join [27–29].

We model each data source as a disjoint set of tuples. Furthermore, to abstract the access model, we assume sampling one tuple at a time with replacement. Although in some settings such as data markets the data is purchased or downloaded in one shot, this query interface is akin to the programming interfaces available in many online services [30]. In particular, these interfaces support selection and count queries, i.e., filtering predicates and integer counts for the number of returned tuples are stated or tuned in the query [31]. Moreover, this assumption is aligned with external data sources, such as web databases, where a limited interface is often enforced that returns a subset of top-k results per query [32–35]. Finally, in a crowd-based data collection setting, workers provide data in smaller batches of tuples.

By default, we assume that the probability change of sampling the same tuple multiple times is negligible. This assumption is reasonable for large data sources and relatively small query requirements. For practical uses, we consider duplicate-aware algorithms as well. In the bulk of the paper, we assume exactly one tuple is returned per query. In § 7, we discuss how our algorithms can be adjusted to relax this assumption.

2.2 Groups

We assume that each data point belongs to one of the disjoint groups $\mathcal{G} = \{G_1, \ldots, G_m\}$ such that $D_1 \cup \ldots \cup D_n = G_1 \cup \ldots \cup G_m$. These groups may be independent, dependent, or omitted variables, as well as subgroups of more than one variable. To illustrate, consider the motivating example in which data scientists are training an ML model to automate the loan approval process. To ensure fair and reliable classification of all groups of interest, data scientists may consider the following minimum count requirements.

- *Independent variables.* For example, all income brackets must be adequately represented.
- *Dependent variables.* For example, manual intervention is required to oversample the minority outcome class due to low loan approval rates [7, 36].
- *Protected variables.* Stepping out of the loan approval example, using models that discriminate based on age or ethnicity for employment decision is illegal in the US [37]. Mitigating disparate impact [38], without also sacrificing accuracy, requires an adequate number of data points for protected groups.
- *Subgroups.* Instead of top-level groups, we may have minimum query count requirements to ensure more than one constraint is met at the same time. Subgroups could also be useful for preventing fairness gerrymandering, where fairness requirements are met for top-level groups but not subgroups [39–41]. For example, the subgroups `white male`, `non-white male`, `white female`, and `non-white female` are used in Example 1. A slice [10] is a synonym for a subgroup.

For the rest the paper, we assume that the number of groups is some constant m. The number of subgroups, however, grows exponentially with respect to the number of top-level groups or features. In order to limit computation time, the maximum depth—the number of top-level groups that a subgroup intersects—may need to be limited, as in [42]. Another way to limit the number of subgroups is to use techniques such as Slice Finder or Sliceline [10, 11] to find only the most problematic slices in terms of model performance. These algorithms also offer parameters to limit computation time.

2.3 Cost model

Obtaining samples from different data sources is not for free. Acquiring samples is associated with a cost either monetary or in the form of computation, memory access, or network access cost. Web database APIs (such as Google Flights), for example, allow a limited number of free queries per day from each IP address or would charge per query while enforcing a top-k interface [32–35]. Similarly, relying on data brokers and data marketplaces may incur monetary costs [14–18]. In survey sampling, amortized cost per query could be charged by digital marketing services or be used to reward respondents. Crowdsourcing services such as Amazon Mechanical Turk [19] are frequently used for data annotation and collection, which charges for labor per query per user. For tabular data sources represented by view, we may need to apply costly pre-processing and sampling steps. The literature on join size estimation and approximate query answering shows that online uniform and independent sampling over join queries requires repetitive sampling with sometimes high reject ratios [28, 29] and non-trivial delay complexity for obtaining successful samples [43]. Furthermore, such costs may vary from one source to another, depending on factors such as the length of join paths, their joinability, statistics of data sets, and matching cost [44].

To generalize across different contexts, we summarize all costs of sampling a source D_i as a positive constant C_i. For the cases where each query returns more than one sample or even the whole source, we can amortize the cost across the number of samples.

2.4 Query model

Our goal is to enable integrating data from multiple sources to construct a target data set. A user query describes a *target data set* with a *target schema*, consisting of a collection of attributes. We abstract the user's query as $[Q_1, \ldots, Q_m]$, a list of nonnegative minimum count requirements. Let O be the collected target data set. We say that a query is satisfied in O when for each group G_j, $|O \cap G_j| \geq Q_j$. We make no assumptions about processing of tuples obtained that exceed the minimum count requirements. They may be discarded or undersampled as needed. Many variants of requirements can be posed, depending on the desired application. We discuss several of these in § 7.

2.5 Summary

Definition 1 (Data Distribution Tailoring Problem) We are given n data sources, m groups, sampling cost for each data source, and query requirements as defined above. We aim to collect a target data set O which satisfies the minimum query counts by querying different data sources in a sequential manner while minimizing the expected total query cost.

We define four variants of DT that lie on a quadrant. First, we categorize whether the discrete probability distribution of groups in each data source is known. This dichotomy is necessary, since in many application settings, such as data in the wild, we may not know much about the data sources. In particular, we may not know the count aggregates for different groups. Solving this variant requires us to learn group distributions for each data source on the fly. Second, we categorize whether the probability of sampling a duplicate already in the unified data set is negligible. Accounting for duplicates requires more bookkeeping which could increase runtime and memory usage. This overhead may be necessary if query requirements are large compared to the size of data sources.

3 Known distribution model

In this section, we consider the DT problem with known group distributions for each data source. We study an optimal dynamic programming solution (§ 3.1) as well as approximation algorithms that utilize previous heuristic (§ 3.2), and our improved heuristic (§ 3.3).

3.1 Dynamic programming

Given the count descriptions $\mathcal{Q} = \{Q_1, \cdots, Q_m\}$, our objective is to find the optimal strategy with the minimum expected cost $F(\mathcal{Q})$. The process of collecting the target data set is a sequence of iterative steps, where at every step, the algorithm chooses a data source, queries it, and keeps the queried tuple if it is not a duplicate. Our first attempt is to develop a dynamic programming (DP) solution.

An optimal source at each iteration minimizes the sum of its sampling cost plus the expected cost of collecting the remaining required groups ($F_j(\mathcal{Q})$), based on its sampling outcome. The dynamic programming analysis evaluates this cost recursively by considering all future sampling outcomes and selecting the optimal source in each iteration accordingly. Using the probabilities of discovering a fresh tuple from each group for every data source D_i, the optimal expected query cost for some query $\mathcal{Q}$ has the following recurrence relation.

$$F(\mathcal{Q}) = \min_{\forall D_i \in \mathcal{D}} \Bigg(C_i + \sum_{\substack{j=1 \\ Q_j > 0}}^{m} P_{i,j} F_j(\mathcal{Q}) + \Big(1 - \sum_{\substack{j=1 \\ Q_j > 0}}^{m} P_{i,j}\Big) F(\mathcal{Q}) \Bigg) \tag{1}$$

Here, $P_{i,j}$ is the ratio of tuples from group G_j in source D_i. To simplify the notation, we have introduced $F_j(\mathcal{Q}) = F(Q_1, \cdots, Q_j - 1, \cdots, Q_m)$. If a sample of G_j is added to the target (because it is fresh and belongs to a group whose count requirement is not fulfilled), the remaining cost for building the target is $F_j(\mathcal{Q})$. Therefore, the term $\sum_{j=1, Q_j>0}^{m} P_{i,j} F_j(\mathcal{Q})$ is the expected cost of the target if we add the current sample to the target. The probability of a sample being redundant is $\big(1 - \sum_{j=1, Q_j>0}^{m} P_{i,j}\big)$ and in this case we will have to pay the cost $F(\mathcal{Q})$.

In our DP algorithm, we assume that for all sources and groups, $|D_i \cap G_j|$ is either zero or sufficiently larger than Q_j. This assumption ensures that the probability of discovering a fresh tuple of a particular not-yet-satisfied group from a data source does not change over different iterations. If $|P'_{i,j} - P_{i,j}| < \epsilon$ for all i, j, then each recursion call of Eq. 1 magnifies multiplicative error of $F_j(\mathcal{Q})$ by at most

$$\max \begin{cases} \frac{1+\epsilon}{1-\epsilon} \left(1 + \frac{|\overline{F_j(\mathcal{Q})} - F_j(\mathcal{Q})|}{F_j(\mathcal{Q})}\right) - \frac{1-\epsilon}{1+\epsilon} \\ \frac{1+\epsilon}{1-\epsilon} - \frac{1-\epsilon}{1+\epsilon} \left(1 - \frac{|\overline{F_j(\mathcal{Q})} - F_j(\mathcal{Q})|}{F_j(\mathcal{Q})}\right) \end{cases}$$

where $\overline{F_j(\mathcal{Q})}$ is an outdated value of $F_j(\mathcal{Q})$. The depth of recursion calls is bounded by $\max Q_j$. Thus, there exists some large $|D_i \cap G_j|$ that ensures $F(\mathcal{Q})$ are within a bounded

Table 3 Specification of our toy problem

	C	G_1	G_2
D_1	1	0.4	0.6
D_2	1	0.1	0.9

factor throughout runtime. The precise bound is not our primary concern, rather the existence of one. If this assumption does not hold, then the tabulation described below may need to be recomputed periodically as probabilities change.

We can solve Eq. 1 using tabulation [45]. If a query has a total query count of $Q+1$, then its expected cost depends only on the subproblems with a total query count of Q. Thus, we start from $Q=0$ and increase the query counts until it reaches the user's query requirement. In addition to memorizing $F(\mathcal{Q})$, we also memorize which data source was optimal under each query by looking at which argument was the smallest in Eq. 1's min term. The initial tabulation takes $O\left(nm\Pi_{i=1}^{m} Q_i\right)$ time and $O\left(\Pi_{i=1}^{m} Q_i\right)$ space. Each iteration takes $O(1)$ time via a single table lookup.

Example 2 Consider sources D_1, D_2 and groups G_1, G_2. Furthermore, consider the costs and conditional probabilities in Table 3.

We would like to collect one tuple from each group, i.e., $\mathcal{Q}=[1,1]$. Trivially,

$$F(0,0)=0.$$

We can also compute

$$F(1,0)=\min\left(\frac{1}{0.4},\frac{1}{0.1}\right)=2.5,$$
$$F(0,1)=\min\left(\frac{1}{0.6},\frac{1}{0.9}\right)=1.111.$$

This gives us

$$F(1,1)=\min\Big(\frac{1+0.4\cdot 1.111+0.6\cdot 2.5}{0.4+0.6},\frac{1+0.1\cdot 1.111+0.9\cdot 2.5}{0.1+0.9}\Big)$$
$$=\min(2.944,3.361)=2.944.$$

We see that D_1 is the optimal data source to sample from, and that the optimal expected cost to satisfy query $(1,1)$ is approximately 2.944.

Figure 1 visualizes $F(Q_1,Q_2)$ for all $Q_1,Q_2\le 1000$. We see that it is a monotonically increasing convex surface, though convexity may not hold in general. Equation 1 satisfies a weaker condition of total convex monotonicity [46], in the sense that $F(a,c)\ge F(b,c) \implies F(a,d)\ge F(b,d)$ for all $a<b$ and $c<d$. This is trivially true since Eq. 1 is monotonically increasing. Total convex monotonicity allows us to utilize convex dynamic programming algorithms which are often faster. However, even convex DP takes n^{th}-degree polynomial time [47], which motivates us to investigate faster heuristic algorithms for general cases with non-trivial number of data sources.

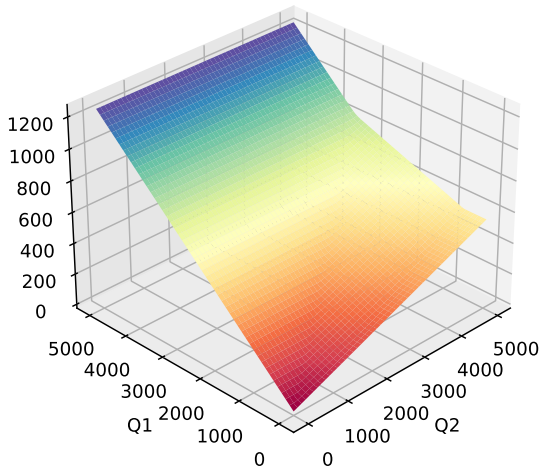


Fig. 1 Surface formed by the optimal cost function applied to Example 2. Color corresponds to the vertical axis: expected cost required to satisfy the query

Algorithm 1 CoupColl

Require: An instance of DT $(\mathcal{D},\mathcal{G},\mathcal{C},\mathcal{Q})$.
Ensure: Unified data set O.
1: $O\leftarrow\emptyset$
2: **while** $\exists j\in[m]$ s.t. $Q_j>0$ **do**
3: $\quad G^*=\underset{G_j\in\mathcal{G},Q_j>0}{\operatorname{argmax}}\left(\min_{i\in[n]}\left(\frac{C_i}{P_{i,j}}\right)\right)$
4: $\quad D^*=\underset{D_i\in\mathcal{D}}{\operatorname{argmin}}\left(\frac{C_i}{P_{i,*}}\right)$
5: $\quad s\leftarrow$ **Query**(D^*)
6: $\quad O\leftarrow O\cup\{s\}$
7: **end while**
8: **return** O

3.2 Approximation algorithm: prior solution

As an alternative to the expensive DP solution, [1] developed an approximation algorithm that models the problem as m instances of *coupon collector problem* [48], where every j^{th} instance aims to collect samples from the group G_j. They also used *union bound* [48] to come up with an upper bound on the expected cost of this CoupColl algorithm.

3.2.1 CoupColl algorithm

Nargesian et al. observed that first collecting data from minority groups helps optimize total cost. This is due to a piggybacking effect, where the chance of collecting data

from other groups while collecting data for minorities is higher than finding minorities while targeting to collect other groups.

Algorithm 1 describes the full CoupColl algorithm. In each iteration, it first identifies the minority group, which is the group for which the most cost-effective data source requires the maximum expected cost. This minority group is defined as

$$G^* = \underset{G_j \in \mathcal{G}, Q_j > 0}{\operatorname{argmax}} \left(\min_{i \in [n]} \left(\frac{C_i}{P_{i,j}} \right) \right). \tag{2}$$

Then, the algorithm samples from the data source which minimizes the expected cost to sample from the minority group. Let $P_{i,*}$ be a shorthand for the probability to sample G^* from D_i. Then,

$$D^* = \underset{D_i \in \mathcal{D}}{\operatorname{argmin}} \left(\frac{C_i}{P_{i,*}} \right). \tag{3}$$

If probabilities $P_{i,j}$ are stationary, then Eqs. 2 and 3 can be pre-computed. As such, CoupColl takes $O(m)$ time per iteration with an upfront $O(nm)$ time and $O(n+m)$ space cost to memorize the values.

3.2.2 CoupCollDupe modification

Unlike our formulation of CoupColl in Algorithm 1, the algorithm as presented in [1] assumed that duplicates cannot simply be ignored. In these cases, Eqs. 2 and 3 must be modified to subsitute $P_{i,j}$ with $P'_{i,j}$, the probability of sampling a non-duplicate tuple of G_j from D_i. To compute $P'_{i,j}$ efficiently, CoupCollDupe keeps track of $N'_{i,j}$, the total number of unique tuples of G_j sampled from D_i. Then, $P'_{i,j} = P_{i,j} - N'_{i,j}/|D_i|$.

It costs $O(nm)$ time once to initialize the trackers and $O(nm)$ space to store them. In each iteration, Eqs. 2 and 3 need to be recomputed, which takes $O(nm)$ time. Uniqueness is tested in $O(1)$ time using a hash set to store O, with size at most Q, the total query count. Overall, CoupCollDupe takes $O(nm+Q)$ space as opposed to $O(n+m)$ space, and $O(nm)$ time per iteration as opposed to $O(m)$ time.

Most other algorithms introduced in the following sections also have duplicate-aware variants, specified with the Dupe suffix. The modifications follow the same method of replacing $P_{i,j}$ with $P'_{i,j}$.

3.2.3 Analysis of CoupColl and CoupCollDupe

Both variants of the CoupColl algorithm always prioritize a group with a nonzero remaining query count. As such, it is at least as efficient as satisfying each group's query requirement independently, then combining all results at the very end. That is, we run m instances of a coupon collector algorithm, where the j^{th} instance repeatedly samples from D^*_j until Q_j is satisfied.

Let C^*_j be the query cost associated with D^*_j. Then, the expected cost to satisfy the query Q_j is $Q_j \,.\, (C^*_j / P_{i,j})$. Thus, if duplicates are negligible, then the expected cost of queries issued by CoupColl, Ψ, is bounded as the following.

$$\Psi \leq \sum_{j \in [m]} \frac{Q_j C^*_j}{P_{i,j}} \tag{4}$$

If duplicates cannot be ignored, then CoupCollDupe is used instead. The same m-parallel-coupon-collectors procedure can be used as an upper bound, though the upper bound is greater due to duplicates. Using the well-known logarithmic expectation to gather a collection of coupons [48], [1] derived the following upper bound.

Theorem 1 *Assume that the probability to get duplicate data points is not negligible. Furthermore, assume that each data source D^*_j for all $j \in [m]$ contains at least Q_j samples from G_j. Then, the expected total query cost issued by* CoupCollDupe *is at most*

$$\sum_{j \in [m]} C^*_j \, |D^*_j| \, \ln \frac{|D^*_j \cap G_j|}{|D^*_j \cap G_j| - Q_j}.$$

Proof Let ψ_j be the number of queries the algorithm would issue to collect Q_j unique tuples from G_j. We note the queries issued to discover the tuples from a group G_j may also discover some tuples from other groups. As a result, the set of queries for different groups may intersect. The union bound [48] indicates that the probability of the union of events is no more than the sum of their probabilities. In DT, the cost of collecting the required tuples of all groups is bounded by the sum of the cost of the tuples of each group. This is because while sampling sources for collecting the next tuple of a particular group, DT keeps the useful tuples of other groups. Using this principle, the expected cost of queries issued by the algorithm, Ψ, is bounded by

$$\Psi \leq \sum_{j=1}^{m} C^*_j \mathbb{E}[\psi_j] \tag{5}$$

For the group G_j, the algorithm queries the data source D^*_j. Let $epoch_j[k]$ be the number of queries issued to collect the k-th tuple of a group G_j. For example, $epoch_j[1]$ is the expected number of queries the algorithm issues until the first tuple from G_j is discovered. Now, if the k-th item from G_j is discovered at the k'-th query, we have $epoch_j[k] =$

$(k' - epoch_j[k-1])$. The number of queries issued at every epoch, ψ_j, is computed as follows.

$$\psi_j = \sum_{k=1}^{Q_j} epoch_j[k]$$

Consider a query that is issued for group G_j to D_j^* during the k-th epoch. Let $P_{j,k}^*$ be the probability that such query is successful, i.e., it discovers a new tuple from G_j. Also let N_j^* be the number of tuples of G_j in D_j^*, i.e., $N_j^* = |D_j^* \cap G_j|$. The algorithm has so far discovered $(k-1)$ tuples and there are $(N_j^* - k + 1)$ undiscovered tuples from G_j at D_j^*. Therefore,

$$\mathbb{P}_{j,k}^* = \frac{N_j^* - k + 1}{N_j^*} \tag{6}$$

The geometric distribution represents the expected number of trials before success in a series of Bernoulli trials. When the probability of discovering a fresh tuple of group G_j is $\mathbb{P}_{j,k}^*$, following the geometric distribution, we have

$$\mathbb{E}\big[epoch_j[k]\big] = \frac{1}{\mathbb{P}_{j,k}^*} \qquad \sigma^2\big[epoch_j[k]\big] = \frac{1 - \mathbb{P}_{j,k}^*}{{\mathbb{P}_{j,k}^*}^2}$$

As a result, we have

$$\begin{aligned}
\mathbb{E}\big[\psi_j\big] &= \mathbb{E}\left[\sum_{k=1}^{Q_j} epoch_j[k]\right] = \sum_{k=1}^{Q_j} \mathbb{E}\big[epoch_j[k]\big] = \sum_{k=1}^{Q_j} \frac{1}{\mathbb{P}_{j,k}^*} \\
&= N_j^* \sum_{k=1}^{Q_j} \frac{1}{N_j^* - k + 1} = N_j^* \sum_{k=(N_j^* - Q_j + 1)}^{N_j^*} \frac{1}{k} \\
&= N_j^* \left(\sum_{k=1}^{N_j^*} \frac{1}{k} - \sum_{k=1}^{N_j^* - Q_j} \frac{1}{k} \right) = N_j^* \big(\mathrm{H}_{N_j^*} - \mathrm{H}_{(N_j^* - Q_j)} \big) \\
&\simeq N_j^* \ln \frac{N_j^*}{N_j^* - Q_j}
\end{aligned}$$

Now, using Eq. 5, we have

$$\Psi = \sum_{j=1}^{m} C_j^* \, N_j^* \ln \frac{N_j^*}{N_j^* - Q_j}$$

□

3.2.4 Binary equi-cost DT revisited

In Theorem 1 of [1], the authors posited that the CoupColl algorithm is optimal under the equi-cost binary-groups constraint. We revisit this result here and argue that while the strategy of selecting the minority first is typically more efficient than the majority strategy, it may incur unnecessary costs for query counts highly skewed toward the majority. First, we restate the theorem as the following proposition.

Proposition 1 *Consider a DT problem under the availability of group distributions where there are two groups and costs for querying data sources are equal. On iteration l, if group $\mathcal{G}_1$ is the minority, i.e., $P_1^* < P_2^*$, then selecting D_1^* is optimal.*

Under the equi-cost binary constraint, CoupColl prioritizes the minority group G_1 until Q_1 is satisfied, at which point it switches to D_2 to prioritize G_2. We can also define the opposite strategy, which prioritizes G_2. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the shorthand for these two algorithms. The following example demonstrates the situation in which $\mathcal{A}_2$ is more efficient than $\mathcal{A}_1$.

Example 3 Consider the toy problem from Example 2 with query requirement $[Q_1, Q_2] = [1, 10]$. We derive an analytical equation for the expected cost of running $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

Consider the event in which j copies of G_2 are sampled while sampling from D_1, with j ranging from 0 to ∞. In such an event, the algorithm samples from D_1, for $(Q_1 + j)$ times, and it samples from D_2, for $\max(Q_2 - j, 0)$ number of times in expectation. Furthermore, the probability of said event is $\binom{Q_1+j-1}{j}(P_1^*)^{Q_1}(1-P_1^*)^j$ since $\binom{Q_1+j-1}{j}$ is the number of combinations for j samples of G_2 and all but the last copy of G_1.

$$\begin{aligned}
\mathbb{E}[\mathcal{A}_1] &= \sum_{j=0}^{\infty} \left(Q_1 + j + \frac{\max(Q_2 - j, 0)}{P_2^*} \right) \\
&\quad \times \binom{Q_1 + j - 1}{j} (P_1^*)^{Q_1} (1 - P_1^*)^j \\
&= \sum_{j=0}^{\infty} (Q_1 + j) \binom{Q_1 + j - 1}{j} (P_1^*)^{Q_1} (1 - P_1^*)^j \\
&\quad + \sum_{j=0}^{\infty} \frac{\max(Q_2 - j, 0)}{P_2^*} \binom{Q_1 + j - 1}{j} (P_1^*)^{Q_1} (1 - P_1^*)^j \\
&= Q_1 \sum_{j=0}^{\infty} \frac{(Q_1 + j)!}{j! Q_1!} (P_1^*)^{Q_1} (1 - P_1^*)^j \\
&\quad + \sum_{j=0}^{Q_2 - 1} \frac{Q_2 - j}{P_2^*} \binom{Q_1 + j - 1}{j} (P_1^*)^{Q_1} (1 - P_1^*)^j \\
&= \frac{Q_1}{P_1^*} + \sum_{j=0}^{Q_2 - 1} \frac{Q_2 - j}{P_2^*} \binom{Q_1 + j - 1}{j} (P_1^*)^{Q_1} (1 - P_1^*)^j
\end{aligned}$$

Algorithm 2 RatioColl

Require: An instance of DT $(\mathcal{D}, \mathcal{G}, \mathcal{C}, \mathcal{Q})$.
Ensure: Unified data set O.
1: $O \leftarrow \emptyset$
2: **while** $\exists j \in [m]$ s.t. $Q_j > 0$ **do**
3: $\quad G^* = \underset{G_j \in \mathcal{G}, Q_j > 0}{\text{argmax}} \left(Q_j \cdot \min_{i \in [n]} \left(\frac{C_i}{P_{i,j}} \right) \right)$
4: $\quad D^* = \underset{D_i \in \mathcal{D}}{\text{argmin}} \left(\frac{C_i}{P_{i,*}} \right)$
5: $\quad s \leftarrow$ **Query**(D^*)
6: $\quad O \leftarrow O \cup \{s\}$
7: **end while**
8: **return** O

Similarly,

$$\mathbb{E}[\mathcal{A}_2] = \frac{Q_2}{P_2^*} + \sum_{j=0}^{Q_1-1} \frac{Q_1 - j}{P_1^*} \binom{Q_2 + j - 1}{j} (P_2^*)^{Q_2} (1 - P_2^*)^j.$$

While this combinatorial equation is impractical to compute for large query counts, we computed $[\mathcal{A}_1]$ and $\mathbb{E}[\mathcal{A}_2]$ for $Q_1, Q_2 \leq 250$ numerically. We find that $\mathbb{E}[\mathcal{A}_2]$ tends to be smaller than $\mathbb{E}[\mathcal{A}_1]$ if Q_2 is well over five times as large as Q_1, with a nearly linear separation between the two regions of the problem space. Neither algorithm is optimal.

As a concrete example, for query $(Q_1, Q_2) = (1, 10)$, $\mathbb{E}[\mathcal{A}_1] = 11.9545$ and $\mathbb{E}[\mathcal{A}_2] = 11.982$, meaning $\mathcal{A}_1$ is more efficient. With just one more query count requirement for G_2, however, $\mathbb{E}[\mathcal{A}_1] = 13.0616$ and $\mathbb{E}[\mathcal{A}_2] = 13.0067$, thus the majority-first algorithm is cheaper. While the minority-first algorithm is typically more efficient than the majority-first algorithm, it may incur unnecessary cost for query counts highly skewed toward the majority.

This example illustrates that the optimal algorithm should balance two priorities at the same time: the minority group and the group with high query count requirement.

3.3 RatioColl: new solution

3.3.1 Modified heuristic

Although CoupColl is suboptimal, the exact solution from § 3.1 is not tractable for even moderately large queries. Consequently, we propose a simple heuristic that is on par or better than CoupColl, which we show empirically in § 5. The modified algorithm RatioColl is shown in Algorithm 2.

RatioColl chooses the priority group based on the expected cost for satisfying the counts of each group independently. Note that C_i / P_{ij} is the expected cost per sample of G_j. Thus, $(Q_j C_j^*)/P_j^*$ is the expected cost to satisfy Q_j on its own, where C_j^* is the cost of D_j^* and P_j^* is the probability of sampling G_j from D_j^*. The following equation is nearly identical to Eq. 2, but with an additional Q_j term.

$$G_k = \underset{j \in [m], Q_j > 0}{\text{argmax}} \left(Q_j \cdot \min_{i \in [n]} \left(\frac{C_i}{P_{i,j}} \right) \right) \tag{7}$$

This modification makes our algorithm balance prioritizing groups that are rare (small $P_{i,j}$), expensive (large C_i), or have a high query count (large Q_j). Once the priority group G^* is chosen, we then choose the optimal data source using Eq. 3. In scenarios where duplicates cannot be ignored, we make an analogous modification as discussed in Sect. 3.2.2.

A natural question is whether assigning weights to the terms Q_j and $\min_{\forall D_i \in \mathcal{D}} \left(\frac{C_i}{P_{i,j}} \right)$ would be necessary. For instance, in a particular scenario, prioritizing a group with a high query count may be much more important than prioritizing a rare group, or vice versa.

We justify our choice by reframing the behavior of RatioColl as follows. Recall that Eq. 7 prioritizes groups based on the expected cost to satisfy the group's query requirement. Thus, at any given point, RatioColl tries to ensure that the expected cost to satisfy a group's query requirement is roughly equal to that of other groups. In other words, RatioColl's goal is when

$$\frac{Q_1 C_1^*}{P_1^*} = \frac{Q_2 C_2^*}{P_2^*} = \ldots = \frac{Q_m C_m^*}{P_m^*}.$$

The sum of all the terms is a constant m times one of the terms.

$$\frac{1}{m} = \frac{\frac{Q_j C_j^*}{P_j^*}}{\sum_{k \in [m]} \frac{Q_k C_k^*}{P_k^*}} = \frac{\frac{Q_j C_j^*}{P_j^*}}{Q \sum_{k \in [m]} \frac{C_k^*}{P_k^*}}$$

Which gives us

$$\frac{Q_j}{Q} = \frac{\frac{P_j^*}{C_j^*}}{m \sum_{k \in [m]} \frac{P_k^*}{C_k^*}}.$$

Normalizing such that Q_j / Q for each group sums to 1, we get an expression for R_j, the proportion of remaining Q_j compared to Q that RatioColl aims to reach.

$$R_j = \frac{\frac{P_j^*}{C_j^*}}{\sum_{k \in [m]} \frac{P_k^*}{C_k^*}} \approx \frac{Q_j}{Q} \tag{8}$$

That is, assuming that the total query count Q is fixed, it wants to have a large proportion of the total query to be occupied by a group that is relatively cheap to obtain. For instance, if G_1 is the minority compared to G_2, then a query

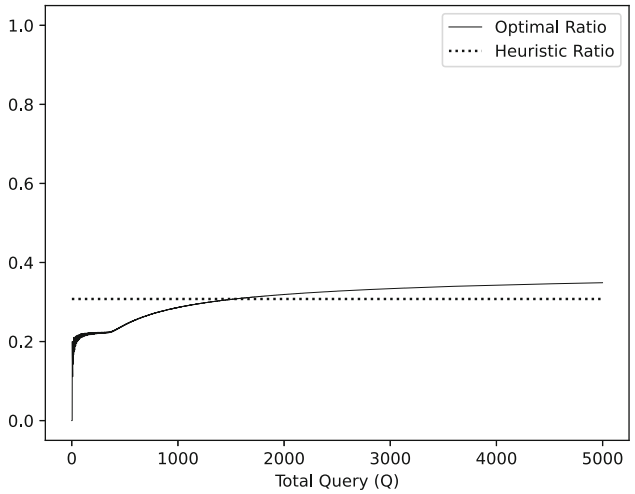


Fig. 2 Optimal $Q_1 : Q_2$ ratio as Q varies

of (20, 80) is easier to satisfy than (80, 20), even though the total query count is the same. It does so by prioritizing the group whose current ratio Q_j/Q is high compared to R_j. RatioColl, over time, brings the ratio of remaining query counts for each group to be closer to the heuristic ratio.

This heuristic ratio is, in many scenarios, remarkably close to the *optimal ratio* which may be computed exactly by computing the DP table. As a concrete example, we computed the exact values of the DP table in Example 2 with decimal point computation to eliminate any numerical issues. Then, for each total query count $0 \leq Q \leq 5000$, we enumerated each possible Q_1, Q_2 combination such that $Q_1 + Q_2 = Q$. We then found the Q_1, Q_2 combination which had the lowest computed $F(Q_1, Q_2)$ value for the given Q. Figure 2 visualizes this result. As Q grows, the optimal ratio approaches 0.35, whereas the heuristic ratio is

$$R_1 = \frac{\frac{0.4}{1}}{\frac{0.4}{1} + \frac{0.9}{1}} \approx 0.31.$$

If Q_1/Q is maintained at 0.31 across the runtime of the algorithm, then while it may not be optimal, it will be close to optimal with dramatically lower runtime compared to DP. Table 4 shows two other combinations of P_1^*, P_2^* under binary-optimal constraint, and the optimal Q_1/Q ratio as Q grows larger compared to the heuristic ratio R_1. We see that the optimal ratio is close to the heuristic in both cases, even though RatioColl is orders of magnitude faster than DP.

3.3.2 Upper bound for RatioColl

Similar to CoupColl, RatioColl only prioritizes a group with nonzero remaining query count. As such, the same loose upper bound shown in Eq. 4 holds. Furthermore, when RatioColl is modified to account for duplicates, the same union bound as shown in Theorem 1 holds.

Table 4 Optimal $\frac{Q_1}{Q}$ when $Q = 5000$ vs R_1

P_1^*	P_2^*	Optimal Q_1/Q Ratio	R_1
0.1	0.95	0.065	0.095
0.5	0.6	0.44	0.45

3.3.3 Asymptotic result

While it is difficult to obtain a tight analysis in general, we show that the expected cost of RatioColl is asymptotically close to optimal under a series of constraints.

Theorem 2 *Consider an instance of DT problem with* $m = 2$, $Q = Q_1 + Q_2$, $Q_1 : Q_2 = P_1^* : P_2^*$, $C_1 = C_2 = 1$, *and* $P_1^* + P_2^* > 1$. *Also assume sampling with replacement from a source which almost surely does not produce duplicates. Recall that* $F(Q_1, Q_2)$ *is the expected cost required to satisfy query* $\mathcal{Q} = (Q_1, Q_2)$. *Then*

$$\lim_{Q\to\infty} \sqrt{Q}[F(Q_1, Q_2) - Q] = \frac{P_1^* + P_2^*}{\sqrt{2\pi\, P_1^* P_2^*}}. \tag{9}$$

The proof amounts to reducing the constrained DT problem into a generalized variant of the coupon collector's problem studied by Brown and Ross [49]. To proceed with the proof, we first restate their problem definition.

Definition 2 [Generalized Coupon Collector's Problem] Suppose there are two types of coupons in a bag. In each iteration, a coupon is drawn randomly with replacement. The probability of drawing coupon 1 is p, and that for coupon 2 is $1 - p$. Our goal is to sample type-1 coupon k times, and type-2 coupon r times. Furthermore, we are given that $p = k/(k+r)$. Estimate the expected time required to satisfy the requirement, denoted $N_{k,r}$.

Proof In order to maintain the ideal ratio, after some t iterations, RatioColl must have sampled tR_1 data points from G_1, and tR_2 data points from G_2. Let X_1^t, X_2^t be the number of data points collected from each group at time t. Since $P_1^* + P_2^* > 1$, as $t \to \infty$,

$$\lim_{t\to\infty} \mathbb{E}\left[\frac{X_1^t}{X_2^t}\right] = \frac{R_1}{R_2}.$$

Consequently, $\lim_{Q\to\infty} \dfrac{F(Q_1, Q_2)}{N_{k,r}} = 1.$

Brown and Ross proved in [49] that

$$\lim_{v\to\infty} \sqrt{v}[\mathbb{E}[N_{k,r}] - v] = \frac{1}{\sqrt{2\pi\, pq}}.$$

Substitute $v = Q$, $N_{k,r} = F(Q_1, Q_2)$, $p = \frac{P_1^*}{P_1^*+P_2^*}$, and $q = \frac{P_2^*}{P_1^*+P_2^*}$ to obtain

$$\lim_{Q\to\infty} \sqrt{Q}[F'(Q_1, Q_2) - Q] = \frac{P_1^* + P_2^*}{\sqrt{2\pi P_1^* P_2^*}}$$

which concludes the proof. □

The significance of Theorem 2 is twofold. First, it is much tighter than Theorem 1. Second, as explained in § 3.3.1, RatioColl tends to stabilize the ratio of remaining query counts to $R_1 : R_2$. Thus, we expect $Q_1 : Q_2 \approx R_1 : R_2$ to be a reasonably common scenario even if the initial query requirements are not exactly as specified in the theorem.

4 Unknown distribution model

In this section, we study the DT problem without knowledge of group probability distributions. A naive solution is to first issue "enough" random queries to each of the data sources and estimate the distributions. Then, knowing these distributions, we can use the techniques proposed in § 3. However, this solution can spend too much of the limited query budget estimating the distributions, especially when there are many data sources or only a small result data set is desired. Therefore, we seek to collect data directly, without first discovering the distributions. To do so, we model the DT problem in the unknown distribution case as a (multi-armed) bandit problem [50–52] (§ 4.1). We then study bandit algorithms for scenarios with (§ 4.2) and without (§ 4.3) prior information.

4.1 Modeling as multi-armed bandit

Multi-armed bandit (MAB) refers to a general class of sequential problems with exploration and exploitation tradeoff. Formally, a stochastic bandit problem is defined as follows. Consider a set of n arms, where each arm a_i is associated with an unknown probability distribution ν_i with mean $\mu(a_i)$. In a sequential setting, with a time horizon T, a planner needs to take action by selecting an arm at every iteration. Let $\mathcal{A} = a_1, \cdots, a_T$ be the sequence of arms chosen by the agent. Upon selecting an arm a_i, the agent receives a stochastic reward $R(a_t)$, from an unknown distribution ν_i with parameter $\mu(a_i)$. We have $\mathbb{E}[R(a_t)] = \mu(a_i)$.

The objective of the agent is to maximize its expected cumulative reward $\mathbb{E}\left[\sum_{t=1}^{T} a_t\right]$. Let the optimal arm at time t be a_t^*. Then, the optimal strategy $\mathcal{A}^* = a_1^*, \cdots, a_T^*$ would have the expected cumulative reward $\sum_{t=1}^{T} \mu(a_t^*)$. Based on this, the *regret* for not taking the optimal actions is computed as follows.

$$\mathcal{L}(T) = \mathbb{E}\left[\sum_{t=1}^{T} \left(\mu(a_t^*) - \mu(a_t)\right)\right] \tag{10}$$

Different strategies have been proposed to balance exploration, which allows the agent to better estimate $\mu(a_i)$, and exploitation, which allows the agent to reap rewards from highly valuable arms. As a naive baseline, we may consider the ε-greedy strategy with a fixed exploration rate ε. In each iteration, with probability $\varepsilon \in [0, 1]$, the planner randomly chooses an arm to explore. Otherwise, it chooses the greedy arm argmax $[\bar{\mu}(a_i)]$, i.e., the arm with the highest sample mean. Each exploration round incurs $O(1)$ regret since it is impossible to optimize random sampling. As such, fixed exploration rate strategies incur $O(T)$ regret.

A sublinear regret bound can be achieved using decreasing exploration rate strategies. These strategies exploit the fact that exploration is more valuable earlier than later, and vice versa. The ε-greedy strategies with ε decreasing over time at an appropriate rate achieves $O\left(T^{2/3} \log T^{1/3}\right)$ regret [52]. This bound can be brought down to $O(T^{1/2} \log T^{1/2})$ through variable exploration rate strategies such as upper confidence bound (UCB) bandit [53].

There is a straightforward mapping of unknown DT problem to stochastic bandit problems, where every data source D_i is an arm. In a sequential manner, we would like to select arms in order to collect Q_j tuples from every group G_j. We still need to design the reward function according to the outcome of a query and the cost of issuing the query, which we explain in the following sections. The design of these reward functions varies depending on the extent to which prior knowledge of the overall distribution is available.

4.2 DT as MAB: with prior knowledge

In order to be able to apply the bandit algorithm, we must define a reward function for each group G_j. In order to compute the reward of collecting a tuple from group $\mathcal{G}_j$, we raise the question of how "hard" it is to collect one tuple of a group. For example, if 90% of the tuples across different data sources belong to G_j, most queries will return a tuple from G_j. On the other hand, collecting a tuple from a group that is rare requires more effort, and so should be worth more in reward. As a result, one can argue that the reward of obtaining a tuple from G_j is proportional to how "rare" this group is across different data sources. In other words, what is the expected cost one needs to pay in order to collect a tuple from G_j.

In order to compute the expected cost, we assume we know the overall distribution of groups as prior information. Such an assumption is reasonable in many scenarios, where overall aggregates are often available in public forms such as Bureau

Algorithm 3 UCB

Require: An instance of DT $(\mathcal{D}, \mathcal{G}, \mathcal{C}, \mathcal{Q})$.
Ensure: Unified data set O.
$O \leftarrow \emptyset$
$N_i \leftarrow 1 \quad \forall i \in [n]$
for $i = 1$ to n **do**
  $s \leftarrow$ **Query**(D_t)
  $\bar{\mu}(D_t) \leftarrow \frac{1}{P^j C_i}$ where $s \in G_j$
end for
while $\exists j \in [m]$ s.t. $Q_j > 0$ **do**
  $D^* \leftarrow \underset{D_i \in \mathcal{D}}{\text{argmax}} \left(\bar{\mu}(D_i) + \sqrt{\frac{2 \ln t}{N_i}} \right)$
  $s \leftarrow$ **Query**(D^*)
  $\bar{\mu}(D_i) \leftarrow \frac{N_1 \cdot \bar{\mu}(D_i) + \frac{1}{P^j C_i}}{N_i + 1}$ where $s \in G_j$
  $N_i \leftarrow N_i + 1$
  $O \leftarrow O \cup \{s\}$
end while
return O

reports. Let P_j be the overall frequency of a group G_j, i.e., $P_j = |G_j|/|\bigcup_{i \in [n]} D_i|$. Following the *principle of deferred decisions* [48] (page 55), if we randomly select a source to query, the expected number of queries required to collect a tuple from G_j is $1/P_j$. Since any source can be selected for sampling, the average cost is $\bar{c} = (\sum_{i=1}^{n} C_i)/n$. Therefore, the expected cost to collect a tuple from G_j is $\bar{c}/P_j$. We would like to assign a high reward to sources that contain tuples of a rare group G_j (small P_j). We also penalize the reward based on the cost of sampling from the source, C_i. Therefore, the reward of source D_i with respect to G_j, namely $R(i, j)$ is $\bar{c}/(P_j \cdot C_i)$. Since $\bar{c}$ is constant across all sources and groups, we remove it from the reward function and write the reward function as following.

$$R(i, j) = \begin{cases} \frac{1}{P^j C_i} & \text{if } Q_j > 0 \text{ and query result is new} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

In order to efficiently compute the average reward of each data source, we keep a running sample mean $\bar{\mu}(D_i)$ for each data source D_i. Once we sample an extra tuple of G_j from D_i, we update the sample mean as

$$\bar{\mu}(D_i) \leftarrow \frac{N_i \cdot \bar{\mu}(D_i) + R(i, j)}{N_i + 1}.$$

Otherwise, Algorithm 3 follows a standard UCB strategy. It requires $O(n+m)$ space to store trackers N_i, $\bar{\mu}(D_i)$ and P_j. Each iteration requires $O(n)$ time. Since Algorithm 3 follows the UCB bandit strategy, and its reward is bounded within a finite range, it incurs $O(\log T)$ regret [52].

Algorithm 4 EPSILONGREEDY

Require: An instance of DT, exploration rate α.
Ensure: Unified data set O.
$O \leftarrow \emptyset$
$N_{i,j} \leftarrow 0 \quad \forall i \in [n], j \in [m]$
while $\exists j \in [m]$ s.t. $Q_j > 0$ **do**
  if $t \leq n$ **then** $D^* \leftarrow D_t$
  else
    $r \leftarrow$ a uniformly random number in [0, 1]
    if $r < \alpha \sqrt[3]{\ln t / t}$) **then**
      $D^* \leftarrow$ a random data source
    else
      $R(G_j) \leftarrow \left(Q_j \cdot \min_{i \in [n]} \left(\frac{C_i N_i}{N_{i,j}} \right) \right) \quad \forall G_j \in \mathcal{G}$
      $D^* \leftarrow \underset{D_i \in \mathcal{D}}{\text{argmax}} \left(\frac{1}{C_i} \sum_{j \in [m]} \frac{N_{i,j}}{N_i} \cdot R(G_j) \right)$
    end if
  end if
  $s \leftarrow$ **Query**(D^*)
  $O \leftarrow O \cup \{s\}$
  $N_{i,j} \leftarrow N_{i,j} + 1$ where $s \in G_j$
  $N_i \leftarrow N_i + 1$
end while
return O

4.3 DT as MAB: without prior knowledge

Although the reward function of equation 11 performs well empirically (§ 5), there are scenarios in which it would not be applicable. Overall demographic statistics may not be available for the problem setting, especially if data points are not individuals. While external statistics could be a substitute, they may deviate significantly from the distribution of given data sources.

4.3.1 Proposed EPSILONGREEDY algorithm

Motivated by the aforementioned limitations, we propose an algorithm for the DT problem with unknown statistics and unknown overall demographics. Algorithm 4 shows the EPSILONGREEDY algorithm. It first samples each data source, then explores a random data source with decrease in probability. Otherwise, it chooses the data source which has the highest expected reward by estimating Eq. 7 from the groups sampled so far from each data source. It exploits further cost optimizations in the same manner as RATIOCOLL by directly utilizing Eq. 7 as the reward function. Although Q_j and C_i can be treated as constants in each iteration, $P_{i,j}$ is unknown by definition. As such, each action does not return a numeric reward, unlike the standard MAB problem definition. Instead, the agent estimates $P_{i,j}$ with increase in accuracy as it samples each data source. This then allows it to estimate Eq. 7 more accurately over time. It requires $O(nm)$ space to store trackers and $O(nm)$ time per iteration.

Notice that the feedback given to EPSILONGREEDY is the group of sampled data point, and not a numerical reward.

Table 5 Terms for analysis of EPSILONGREEDY

Symbol	Description
X^t	Number of exploration rounds by time t
X_i^t	Number of times D_i was explored by time t
$Y_{i,j}^t$	Number of times G_j was sampled from D_i by time t
$P_{i,j}$	Probability of sampling G_j from D_i
$R(G_j)$	Reward from group G_j, as in equation 7 i.e., $R(G_j) = Q_j \frac{C_j^*}{P_j^*}$
$\mu(D_i)$	Expected reward from D_i i.e., $\mu(D_i) = \sum_{j\in[m]} P_{i,j} R(G_j)$
$\mathcal{L}(t)$	Regret by time t i.e., $\mathcal{L}(t) = \sum_{t=1}^{T} f(D^*) - \mathbb{E}[f(D_t^*)]$ where D^* is the optimal source, and D_t^* is the source chosen by the algorithm at time t

This is the main deviation from the standard bandit problem definition (§ 4.1). As such, we cannot rely on prior results. In designing a bandit algorithm for this modified problem definition, we could consider a UCB strategy. However, this is not feasible, as the reward of each arm is non-local. Furthermore, the reward of each arm is non-local. In order to accurately estimate the reward from D_i, we need to explore *all* data sources, not just D_i. This limitation is why we opt for an ε-greedy strategy with ε that decays over time.

4.3.2 Analysis of EPSILONGREEDY

Surprisingly, despite an additional layer of statistical inference, our EPSILONGREEDY algorithm has a big-O bound identical to ε-greedy bandit algorithms in the standard bandit problem setting.

Theorem 3 *The* EPSILONGREEDY *algorithm achieves asymptotic regret of* $O\left(t^{2/3}(\log t)^{1/3}\right)$ *at time* t, *when* $C_i = 1$ *for all* $i \in [n]$, n, m, Q_j *are fixed, and* p^* *is the minimum optimal probability for any group defined as* $p^* = \min_{j\in[m]}\left[\max_{i\in[n]}\left(P_{i,j}\right)\right] > 0$.

Notice that Theorem 3 assumes all Q_j to be constants. This is a limitation imposed by the fact that remaining query counts at a certain iteration is unpredictable. However, once the query counts converge to the ratio imposed by Eq. 8, we argue that the remaining query count for a group relative to all remaining queries will stay mostly constant throughout the rest of the algorithm's runtime with high probability. As such, fixing Q_j as constant is a reasonable simplifying assumption that does not significantly impact the implications of the theorem.

To proceed with the proof, we define some new terms and recall others in Table 5. Another concept we must define is the *clean event*, which occurs when all values of X^t, X_i^t, and $Y_{i,j}^t$ are not far from expectation.

Definition 3 We define events $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ as the events in which equations 12, 13, or 14 hold, respectively.

$$\left|X^t - \mathbb{E}[X^t]\right| \le O\left(t^{-1/3}(\log t)^{-1/6}\right)\mathbb{E}[X^t] \tag{12}$$

$$X_i^t \ge \mathbb{E}\left[X_i^t\right] - O\left(t^{-1/3}(\log t)^{-1/6}\right)\mathbb{E}[X_i^t] \quad \forall i \in [n] \tag{13}$$

$$Y_{i,j}^t \ge \mathbb{E}\left[Y_{i,j}^t\right] - O\left(t^{-1/3}(\log t)^{-1/6}\right)\mathbb{E}[Y_{i,j}^t] \quad \forall i \in [n]\, \forall j \in [m] \tag{14}$$

The **clean event** $\mathcal{E}_c$ is the event in which all three equations hold, i.e., $\mathcal{E}_c = \mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$. The complement of the clean event is called the *dirty event*, denoted $\overline{\mathcal{E}_c}$.

Finally, any term with a horizontal bar above it (e.g., $\bar{P}_{i,j}$, $\bar{\mu}(D_i)$) is an estimate of the ground truth obtained through sampling.

The proof strategy is straightforward: we bound the terms in Table 5 one by one with high probability using standard concentration bounds. For organization, the proof is split into lemmas. Lemma 1 shows that if the clean event holds, then $\mu(D_i)$ is closely estimated. Lemma 2 bounds regret given the clean event. Lemma 3 bounds probability of the dirty event. From these lemmas, Theorem 3 follows directly.

Lemma 1 *If the clean event holds, then*

$$|\bar{\mu}(D_i) - \mu(D_i)| \le O\left(t^{-1/3}(\log t)^{-1/6}\right)\mu(D_i).$$

Proof Since the clean event holds, as a direct application of Eq. 13 and Eq. 14,

$$|\bar{P}_{i,j} - P_{i,j}| \le O\left(t^{-1/3}(\log t)^{-1/6}\right)P_{i,j}.$$

As a shorthand, let $\delta = t^{-1/3}(\log t)^{-1/6}$.

The bound on $|\bar{P}_{i,j} - P_{i,j}|$ implies a similar bound on $|\bar{R}(G_j) - R(G_j)|$.

$$\begin{aligned}\left|\bar{R}(G_j) - R(G_j)\right| &\le \max\begin{cases}\frac{1}{(1-\delta)P_{*,j}} - \frac{1}{P_{*,j}} \\ \frac{1}{P_{*,j}} - \frac{1}{(1+\delta)P_{*,j}}\end{cases} \\ &\le \frac{1}{(1-\delta)P_{*,j}} - \frac{1}{(1+\delta)P_{*,j}} \\ &= \frac{2\delta}{(1-\delta^2)}\frac{1}{P_{*,j}}\end{aligned}$$

$$= \frac{O\left(t^{-1/3}(\log t)^{-1/6}\right)}{\Omega(1)} R(G_j)$$
$$\therefore |\bar{R}(G_j) - R(G_j)| \leq O\left(t^{-1/3}(\log t)^{-1/6}\right) R(G_j)$$

We obtain a similar bound for $|\bar{\mu}(D_i) - \mu(D_i)|$ by repeating the same technique.

$$\begin{aligned}
&|\bar{\mu}(D_i) - \mu(D_i)| \\
&= \left| \sum_{j\in[m]} \bar{P}_{i,j}\bar{R}(G_j) - \sum_{j\in[m]} P_{i,j}R(G_j) \right| \\
&= \sum_{j\in[m]} \left[\left|\bar{P}_{i,j}\bar{R}(G_j)\right| - \left|P_{i,j}R(G_j)\right| \right] \\
&\leq \sum_{j\in[m]} \left|\bar{P}_{i,j}\bar{R}(G_j) - P_{i,j}R(G_j)\right|
\end{aligned}$$

The summand is simplified as follows.

$$\begin{aligned}
&\left|\bar{P}_{i,j}\bar{R}(G_j) - P_{i,j}R(G_j)\right| \\
&\leq \max \begin{cases} (1+\delta)P_{i,j}(1+\delta)R(G_j) - P_{i,j}R(G_j) \\ P_{i,j}R(G_j) - (1-\delta)P_{i,j}(1-\delta)R(G_j) \end{cases} \\
&\leq [(1+\delta)^2 - (1-\delta)^2]P_{i,j}R(G_j) \\
&= 4\delta P_{i,j}R(G_j) \\
&= 4\delta R(G_j) \\
&\therefore |\bar{\mu}(D_i) - \mu(D_i)| \leq O\left(t^{-1/3}(\log t)^{-1/6}\right)\mu(D_i)
\end{aligned}$$

□

Lemma 2 *If the clean event holds, then*

$$\mathcal{L}(t) \leq O\left(t^{2/3}(\log t)^{1/3}\right).$$

Proof Similar to the proof of lemma 1, let $\delta = t^{-1/3}(\log t)^{-1/6}$ as a shorthand.

Regret can incur during exploration rounds and during exploitation rounds. Consider the exploration rounds first. Since the clean event holds, by equation 12, X^t is upper bounded by

$$X^t \leq (1+\delta)\mathbb{E}[X^t] \leq \left(1 + O\left(t^{-1/3} + (\log t)^{-1/6}\right)\right)\mathbb{E}[X^t].$$

Since the exploration rate is $\sqrt[3]{\ln t/t}$, by integration and algebraic simplification, $\mathbb{E}[X^t] = \Theta\left(t^{2/3}(\log t)^{1/3}\right)$. Thus, by substitution,

$$X^t \leq O\left(t^{2/3}(\log t)^{1/3}\right).$$

Each iteration of exploration round incurs at most $O(1)$ regret, since $p^* > 0$ implies that the maximum possible reward is bounded. As such, the regret from exploration rounds is at most $O\left(t^{2/3}(\log t)^{1/3}\right)$.

Now consider the regret from exploitation rounds. There is at most $t - (1-\delta)\mathbb{E}[X^t] \leq O(t)$ exploitation rounds. For each exploitation round, if a suboptimal source D_i was chosen instead of the optimal source D^*, it must be that

$$\begin{aligned}
&\left[1 + O\left(t^{-1/3}(\log t)^{-1/6}\right)\right]\mu(D_i) \\
&\quad \geq \left[1 - O\left(t^{-1/3}(\log t)^{-1/6}\right)\right]\mu(D^*)
\end{aligned}$$

which means

$$\mu(D_i) \geq \frac{1 - O\left(t^{-1/3}(\log t)^{-1/6}\right)}{1 + O\left(t^{-1/3}(\log t)^{-1/6}\right)}\mu(D^*).$$

Maximum reward in any source is $1/p^*$, so

$$\begin{aligned}
\mu(D^*) - \mu(D_i) &\leq \frac{1}{p^*} - \frac{1 - O\left(t^{-1/3}(\log t)^{-1/6}\right)}{1 + O\left(t^{-1/3}(\log t)^{-1/6}\right)}\frac{1}{p^*} \\
&= \frac{1}{p^*}\left(1 - \frac{1 - O(t^{-1/3}(\log t)^{-1/6})}{1 + O(t^{-1/3}(\log t)^{-1/6})}\right) \\
&= \frac{1}{p^*}\left(\frac{2O(t^{-1/3}(\log t)^{-1/6})}{1 + O(t^{-1/3}(\log t)^{-1/6})}\right) \\
&= O(t^{-1/3}(\log t)^{-1/6}).
\end{aligned}$$

Thus, the regret from exploitation rounds is at most $O(t^{-1/3}(\log t)^{-1/6})$.

Combining the regret from exploration rounds and exploitation rounds,

$$\begin{aligned}
&\mathcal{L}(t) \leq O(t^{2/3}(\log t)^{1/3}) + O(t^{-1/3}(\log t)^{-1/6}), \\
&\therefore \mathcal{L}(t) \leq O(t^{2/3}(\log t)^{1/3}).
\end{aligned}$$

□

Lemma 3 *The dirty event occurs with probability at most* $O(t^{-4})$.

Proof Let us denote the event in which Eq. 12 holds as $\mathcal{E}_1$, the event in which Eq. 13 holds as $\mathcal{E}_2$, and the event in which Eq. 14 holds as $\mathcal{E}_3$. Then, the clean event $\mathcal{E}_c = \mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$, so $P(\mathcal{E}_c) = P(\mathcal{E}_1)P(\mathcal{E}_2|\mathcal{E}_1)P(\mathcal{E}_3|\mathcal{E}_1 \cap \mathcal{E}_2)$.

First, we lower bound $P(\mathcal{E}_1)$ with multiplicative Hoeffding bound.

$$\mathbb{P}\left[|X^t - \mathbb{E}[X^t]| \geq \delta\mathbb{E}[X^t]\right] \leq 2e^{-\delta^2\mathbb{E}[X^t]/3}.$$

We set $\delta = 2t^{-1/3}(\log t)^{-1/6}$ such that Eq. 12 holds. Then by substitution,

$$2e^{-\delta^2\mathbb{E}[X^t]/3} = \Theta\left(t^{-4}\right),$$

so $P(\mathcal{E}_1) \geq 1 - O(t^{-4})$.

Second, we lower bound $P(\mathcal{E}_2|\mathcal{E}_1)$. Since event $\mathcal{E}_1$ holds, by substitution,

$$X^t \geq (1-\delta)\mathbb{E}[X^t],$$
$$X^t \geq \left(1 - 2t^{-1/3}(\log t)^{-1/6}\right) \Theta\left(t^{2/3}(\log t)^{1/3}\right).$$

After simplification, we get

$$\Theta\left(t^{2/3}(\log t)^{1/3}\right) \leq X^t.$$

Each X_i^t is a sum of independent random variables. So again, by the multiplicative Hoeffding bound,

$$\mathbb{P}\left[X_i^t \geq (1-\delta)\mathbb{E}[X_i^t] \mid \mathcal{E}_1\right] \leq e^{-\delta^2 \mathbb{E}[X_i^t]/2}.$$

Setting $\delta = 2t^{-1/3}(\log t)^{-1/6}$ ensures that Eq. 13 holds. By substitution, $P(\mathcal{E}_2|\mathcal{E}_1) \geq 1 - O(t^{-4})$.

Then by a similar process as above,

$$\mathbb{P}\left[\left|Y_{i,j}^t - \mathbb{E}[Y_{i,j}^t]\right| \geq \delta\mathbb{E}[Y_{i,j}^t] \mid \mathcal{E}_1 \cap \mathcal{E}_2\right] \leq \Theta(t^{-4})$$

where $\delta = t^{-1/3}(\log t)^{-1/6}$. This means $P(\mathcal{E}_3|\mathcal{E}_1 \cap \mathcal{E}_2) \geq 1 - O(t^{-4})$.

Combining the above results,

$$\begin{aligned} P(\mathcal{E}_c) &= P(\mathcal{E}_1)P(\mathcal{E}_2|\mathcal{E}_1)P(\mathcal{E}_3|\mathcal{E}_1 \cap \mathcal{E}_2) \\ &\geq \left(1 - O(t^{-4})\right)\left(1 - O(t^{-4})\right)^n \left(1 - O(t^{-4})\right)^m \\ &\geq 1 - O(t^{-4}). \end{aligned}$$

Thus, $P\left(\overline{\mathcal{E}_c}\right) \leq O(t^{-4})$. □

Now we may prove Theorem 3.

Proof The regret per iteration given the dirty event is $O(1)$ by the same reasoning as why regret per exploration round is $O(1)$. Then, we have

$$\begin{aligned} \mathcal{L}(t) &\leq O\left(t^{2/3}(\log t)^{1/3}\right) + O(t^{-4})O(1) \\ &= O\left(t^{2/3}(\log t)^{1/3}\right) \end{aligned}$$

which completes the proof. □

4.3.3 A practical variant

Although EPSILONGREEDY has the best theoretical regret bound at arbitrary time t, an EXPLOREEXPLOIT strategy described in Algorithm 5 could be much more efficient in practice. This is because the expected time horizon is at most some unknown constant multiple of Q, as established in Eq. 4. As such, the time horizon can be crudely approximated as $T \approx Q$, and the exploration rate set to $\alpha T^{2/3}$ for some tunable parameter α. Given a known time horizon, it is highly advantageous to perform all explorations at the beginning. Furthermore, EXPLOREEXPLOIT's simplicity means the exploration rounds can be batched and parallelized, and an existing implementation of RATIOCOLL can be reused as a subroutine. Although there is no firm theoretical bound, if the time horizon is accurately estimated, then by a direct application of Theorem 3, EXPLOREEXPLOIT achieves the same sublinear regret as EPSILONGREEDY.

Algorithm 5 EXPLOREEXPLOIT

Require: An instance of DT, exploration rate α.
Ensure: Unified data set O.
1: $O \leftarrow \emptyset$
2: $N_{i,j} \leftarrow 0 \quad \forall i \in [n], j \in [m]$
3: **for** $i = 1$ to $\lceil \alpha Q^{2/3} \rceil$ **do**
4: $\quad s \leftarrow$ **Query**$(D_{(i \mod n)+1})$
5: $\quad N_{i,j} \leftarrow N_{i,j} + 1$ where $s \in G_j$
6: $\quad N_i \leftarrow N_i + 1$
7: **end for**
8: Run RATIOCOLL with $\bar{P}_{i,j} = \frac{N_{i,j}}{N_i}$

5 Experiments

We have developed the RATIOCOLL algorithm in this paper for the case of known distributions, which improves upon COUPCOLL from [1]. We also developed UCB for the case of unknown distributions with prior information, and EPSILONGREEDYand EXPLOREEXPLOIT for the case of no prior information. We empirically study the performance of the algorithms and compare them against baselines.

We use a RANDOM baseline that chooses a random data source in each iteration. For the unknown scenario, we also consider a FIXEDEXPLORATION baseline which uniformly explores for 10% of total query Q then runs *RatioColl* using sample statistics. For each algorithm, their duplicate-aware variants are denoted by the DUPE suffix, replacing $P_{i,j}$ with $P'_{i,j}$ as discussed in Sect. 3.2.2.

We design a series of experiments that verify our theoretical results under a range of problem sizes and test the robustness of our proposed algorithms under a variety of scenarios. In total, we conduct six experiments as listed below.

1. Comparing the performance of algorithms while varying the proportion of the query count requirement occupied by minority versus majority group. (Sect. 5.2.1, Fig. 3)
2. Comparing RATIOCOLL and its theoretical bounds (Theorem 1,2) while varying Q to test the practical usefulness of analysis. (Sect. 5.2.2, Fig. 4a)

3. Comparing FIXEDEXPLORATION baseline with decaying exploration rate bandits EXPLOREEXPLOIT and EPSILON-GREEDY while varying Q. (Sect. 5.2.3, Fig. 4b)
4. Testing the proposed algorithms on a large practical data set. (Sect. 5.2.4, Fig. 4c)
5. Testing the proposed algorithms and their duplicate-aware variants on a small practical data set. (Sect. 5.2.5, Fig. 4d)
6. Comprehensive performance evaluation while varying cost model, n, m, and whether each group has a source where it is the majority group. (Sect. 5.2.6, Figs. 5,6)

5.1 Data

Flights [54]: Airborne Flights database, published by the Bureau of Transportation Statistics, contains detailed flight statistics from 1987 to present. The carrier on-time performance of each flight is represented by `OP_CARRIER_AIRLINE_ID`, `ORIGIN_STATE_NM`, and `ARR_DELAY`, among other attributes. We considered the flight information of carrier airlines from 2018 to 2020, each row representing a single flight. There were 18 million flights in total. The data set was split into 11 data sources, each representing an airline network. We categorize data points into 51 groups by their destination state, including U.S. Pacific territories.

COMPAS [55]: The Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) data set was used to demonstrate racial bias in the COMPAS recidivism algorithm [56]. It is now used as an example of a biased data set [41, 57, 58]. We deduplicated rows based on an individual's ID, resulting in a data set of 18,610 individuals. We split the data set based on the month of their screening, with a total of 24 data sources over two years. We categorized each individual into demographic groups: `Caucasian Male`, `Non-Caucasian Male`, `Caucasian Female`, and `Non-Caucasian Female`. The largest group `Caucasian Male` had 9,407 tuples, and the smallest group `Non-Caucasian Female` had 1,706.

BenchDL: We synthesized a benchmark to evaluate DT on various cost and data distribution settings. It can generate n random synthetic data sources with m groups, where each data source generates unique tuples on the fly according to an underlying probability distribution. *BenchDL* has two modes of randomly generating the underlying probability distributions: majority and minority. Majority distribution means that for all group G_j, $P_j^* \geq 1/m$, whereas a minority distribution means that for at least one group, $P_j^* \leq 1/m$. *BenchDL* generates majority distributions by allocating a probability of $1/m$ to each group in at least one data source, then randomly splitting remaining probabilities. It generates minority distributions by allocating a probability that lies in $(0, 1/m)$ to all data sources to at least one group, then randomly splitting remaining probabilities.

BenchDL assigns a constant cost to each data source, chosen by one of three cost models.

1. *Uniform*: $C_i = 1$ for all $i \in [n]$.
2. *Random*: A random floating point number in $(0, 2]$.
3. *Skewed*: We generate a number from the Pareto distribution with parameter $\alpha = 2$, then subtract 1 to obtain a number in range $(0, \infty)$.

The parameters are set such that $\mathbb{E}[C_i] = 1$ for all cost models, which means the expected cost per iteration for the RANDOM baseline is 1. This property allows us to discern which algorithms better exploit the cheaper data sources.

5.2 Experiment design & results

5.2.1 Experiment 1: query ratio under equi-cost binary constraint

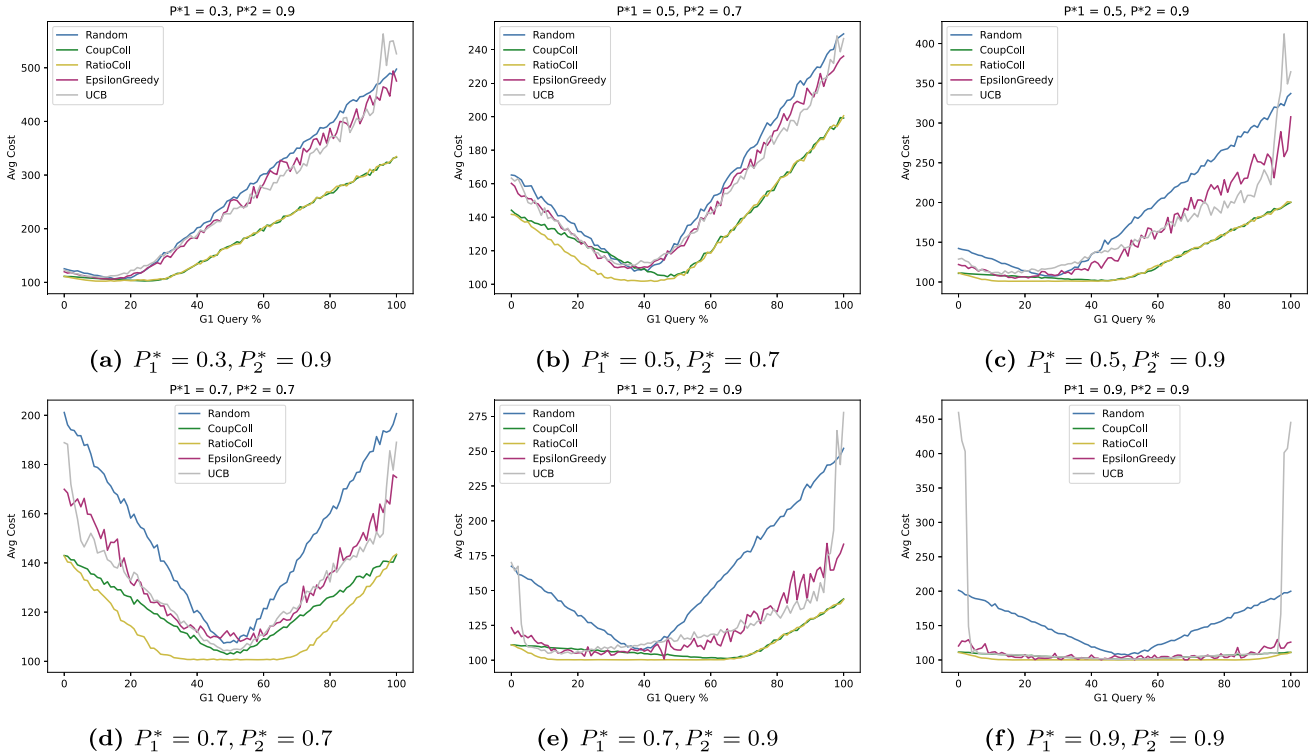
The goal of this experiment is to compare the performance of the discussed algorithms under an equi-cost binary constraint with various P_1^*, P_2^* combinations and $Q_1 : Q_2$ ratios. We set $n = 2, m = 2$ and use the uniform cost model. Furthermore, in all rounds, we set the total query count $Q = 100$. There are three independent variables: Q_1, P_1^* and P_2^*. We vary P_1^* and P_2^* among values 0.1, 0.3, 0.5, 0.7, and 0.9. Since there are only two data sources, combinations where $P_1^* + P_2^* < 1$ are infeasible, and combinations where $P_1^* + P_2^* = 1$ result in identical D_1 and D_2. Thus, we only consider $P_1^* + P_2^* > 1$. Some combinations are omitted due to symmetry. We vary Q_1 from 0 to 100.

The results are shown in Fig. 3. In the known setting, COUPCOLL and RATIOCOLL consistently outperforms the RANDOM baseline. Furthermore, there are regions of Q_1 values where COUPCOLL consistently outperforms RATIOCOLL, particularly when Q_1 is relatively small. This effect is most evident in Fig. 3(d-f). This is to be expected, since we always set G_1 to be the minority group; COUPCOLL is inefficient when the minority group has small query requirements.

Though variance is high in the unknown setting, bandit strategies tend to outperform RANDOM with the exception of UCB. While UCB tends to outperform RANDOM, it may, in certain configurations, be worse than the RANDOM baseline. UCB's lackluster performance in certain scenarios confirms our claim that the reward function for UCB, even if known, may not be applicable.

5.2.2 Experiment 2: doubling test for bounds

In this experiment, we test the applicability of analysis on RATIOCOLL for reasonably small query counts under the conditions of Theorem 2. Specifically, we set $n = 2, m = 2$, $P_1^* = 0.5$, and $P_2^* = 0.75$ with uniform query costs. As



(a) $P_1^* = 0.3, P_2^* = 0.9$ (b) $P_1^* = 0.5, P_2^* = 0.7$ (c) $P_1^* = 0.5, P_2^* = 0.9$

(d) $P_1^* = 0.7, P_2^* = 0.7$ (e) $P_1^* = 0.7, P_2^* = 0.9$ (f) $P_1^* = 0.9, P_2^* = 0.9$

Fig. 3 Performance of all algorithms under equi-cost binary constraint with varying $Q_1 : Q_2$ ratio, and $P_1^* + P_2^* > 1$. P_1^*, P_2^* was chosen from 0.3, 0.5, 0.7, 0.9

the independent variable, we vary the total query count Q from 32 to 5,096 as a doubling test. We also ensure that $Q_1 : Q_2 \approx 0.5 : 0.75$ to the nearest integer regardless of Q. We run the RANDOM and RATIOCOLL algorithms, and compare their results with the union bound (Theorem 1) and the asymptotic expectation (Theorem 2). For each query count and algorithm, we repeat for 30 repetitions and take the average.

The result is shown in Fig. 4a. Even with a relatively small query count requirement in the thousands, the asymptotic expectation is very close to the average cost issued by RATIOCOLL to satisfy the query. Furthermore, union bound is loose compared to the asymptotic expectation, far surpassing even the random baseline.

5.2.3 Experiment 3: fixed versus uniform exploration strategies

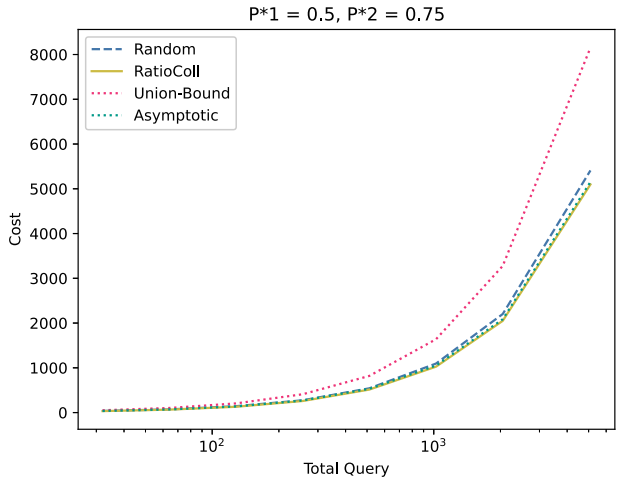
In this experiment, we compare the FIXEDEXPLORATION baseline with our proposed decaying exploration rate bandit algorithms EPSILONGREEDY and EXPLOREEXPLOIT ($\alpha = 0.5$) as query size Q increases. We set $n = 5, m = 10$, with no majority groups and skewed cost model. We then doubled Q from 2^4 to 2^{14}.

The result is shown in Fig. 4b. It plots the average cost for each strategy per Q divided by the random baseline. As Q increases, the bandit strategies trend toward significantly outperforming the random baseline. Out of the three bandits, EXPLOREEXPLOIT is the clear winner. Intriguingly, EPSILONGREEDY performs worse than the baselines for certain values of Q but nonetheless improves in an asymptotic manner.

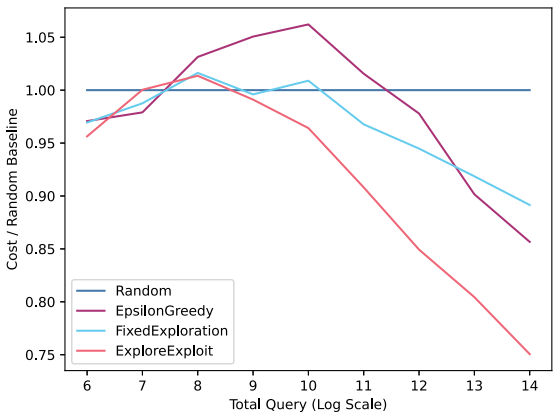
5.2.4 Experiment 4: real world, no duplicates

We run the COUPCOLL, RATIOCOLL, UCB, EPSILONGREEDY, and RANDOM algorithms on the flights data set, with the group count requirements set as 100 per group. Thus, $Q = 5,100$ which is less than 0.3% of all data. Since Q is small compared to the data sources, we ignore the effect of duplicates. We also assume a uniform cost model, since all data sources come from the same publisher. We report the average over 10 rounds.

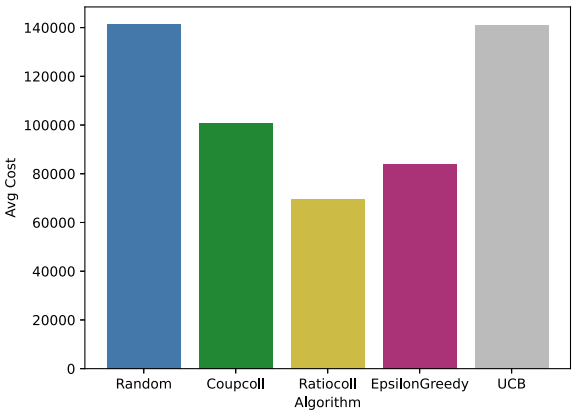
The result is shown in Fig. 4c. RATIOCOLL outperforms COUPCOLL, which out-perofrms the RANDOM baseline. Futhermore, the query cost issued by EPSILONGREEDY is competitive to RATIOCOLL, especially in contrast to UCB that does not outperform RANDOM. The probability distributions of the flight data set might be one where UCB's heuristic is not the most applicable.



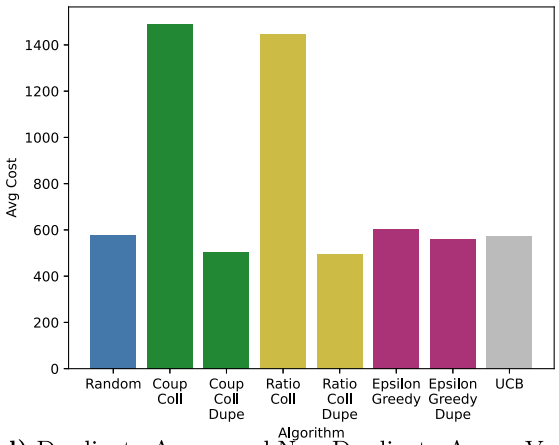
(a) Comparing RatioColl and theoretical bounds under equi-cost binary as Q grows.



(b) Varying Q for uniform-exploration bandit strategies.



(c) Proposed algorithms tested on Flights dataset.



(d) Duplicate-Aware and Non-Duplicate-Aware Variants on COMPAS Dataset.

Fig. 4 Results of Experiment 2–5

5.2.5 Experiment 5: real world, duplicates

We run the CoupColl, CoupCollDupe, RatioColl, RatioCollDupe (a natural duplicate-aware variant of RatioColl), UCB, EpsilonGreedy, EpsilonGreedyDupe, and Random algorithms on the COMPAS data set. The smallest $P_{i,j}$ value is around 50, so we set each group's query requirement as 50 to ensure that duplicates are non-negligible. We report the average over 50 rounds.

The result is shown in Fig. 4d. Since the data sources are very small compared to the query requirement, non-duplicate-aware algorithms are highly inefficient compared to even just the Random baseline or their duplicate-aware variants. CoupCollDupe and RatioCollDupe seems to outperform the Random baseline by a small margin. EpsilonGreedy fares much better, even though it is not duplicate-aware. This may be due to the fact that the total query requirement is very small, so exploration rate is still high.

5.2.6 Experiment 6: synthetic, general case

In this experiment, we run all algorithms under the following combinations of independent variables.

1. Cost model: We test all three cost models in *BenchDL* , namely uniform, random, and skewed.
2. Varying n: In half of all experiments, we fix $m = 10$ and vary n as 2, 4, 6, 8, or 10.
3. Varying m: In the other half of all experiments, we fix $n = 10$ and vary m as 10, 20, 30, 40, or 50.

We set n, m values such that $n \leq m$ since the non-duplicate-aware algorithms sample from at most m data sources. As such, scenarios in which $n > m$ are redundant. Furthermore, we split the results into two sets of plots for both known and unknown distributions. The total query count is fixed at 10, 000 and the group count requirements are evenly distributed among groups. We repeat each combination of independent variables 25 times and report the average.

The results of Experiment 5 are shown in Figs. 5 and 6, split into known and unknown DT. In known DT, RatioColl tends to outperform CoupColl, which outperforms the Random baseline. In unknown DT, while UCB and EpsilonGreedy tend to outperform the Random baseline, there is no clear winner between the two bandit algorithms. In certain scenarios, the heuristic for UCB fits well, in which case the $O(\log t)$ regret of UCB is highly desirable over the $O(t^{2/3} \log t)$ regret of EpsilonGreedy. On the other hand, if the reward function for UCB is not quite applicable, then EpsilonGreedy may outperform UCB.

As a general trend, the cost difference between the Random baseline and our proposed algorithms is the smallest for uniform cost model, and the largest for skewed cost model. As all proposed algorithms take the cost of data sources into account, they are adept at exploiting the cheap outlier sources.

Increasing the number of groups tends to increase the cost required to satisfy the query, since rare groups become much harder to find. EpsilonGreedy is affected particularly hard by this trend. On the other hand, increasing the number of data sources tends to decrease the total cost. With many data sources to choose from, our algorithms are better able to exploit the outlier data sources which have highly desirable probability distributions.

5.3 Summary of results

In the experiments, we demonstrated the effectiveness of our proposed algorithms over the Random baseline and algorithms proposed in [1]. We list the major takeaways below.

1. RatioColl outperforms CoupColl when minority group's query count requirements are small.
2. Decaying exploration rate bandits are superior to a fixed exploration baseline as Q increases.
3. ExploreExploit may be superior over EpsilonGreedy in practice for many scenarios due to exploring at the beginning.
4. Duplicate-aware modification is effective in high-duplicate settings.
5. As the variance in costs increases, the gap between the Random baseline and our algorithms increases.
6. With higher group and source counts, the gap between the Random baseline and our algorithms increases.

6 Related work

Responsible Data Science The bulk of work in algorithmic fairness and responsible data science has been on building fair ML models [59]. At a high level, the interventions to achieve fairness in ML fall in three major categories [60]: pre-process techniques [38, 61–63], algorithm modification (in-process) [64–67], and post-process techniques [68–70] that change model outcomes. Alongside other communities, fairness has been a central topic in the premier database research. Related work on data management for algorithmic fairness includes data repair [63, 71], ranking [72–75], and data/model annotation [76, 77], as well as different keynotes [78, 79] and tutorials [80–82].

Bias and Representativeness in Data Biases have been studied for a long time in statistics community [83] but social data presents different challenges [59, 84, 85]. For social data, the term bias refers to demographic disparities in the sampled data that compromises its representativeness and are objectionable for societal reasons [59, 84]. Given that "*an algorithm is only as good as the data it works with*" [85], fairness-aware data collection is considered as a way to address unfairness in predictive models [86, 87]. Representativeness of data collection have been widely studied in the literature [88]. A notion of data representativeness has been proposed as *data coverage* [42, 89–92], identifying the demographic subgroups that are not represented in data. The input target distribution to a DT problem can be inferred from the result of coverage analysis. Bias has also been studied in the context of approximate query answering [93], where a database is considered as a sample and the goal is to answer approximate queries as if the queries were issued on the true population.

Data Discovery and Data Pricing Existing approaches for data set discovery [20, 94], source selection [95, 96], and schema mapping [97–99] can be necessary for the source generation step of DT and their cost can be folded into the cost model. Data set discovery is often formulated as a search problem on repositories using keywords [100, 101] or another data set [20, 94] and the goal is to find relevant data sets based on the relevance to the keywords or integration-inspired measures. A complementary problem to DT is query-based data pricing [102] which decides the price of the data from the perspective of providers. The output of the data pricing problem can be plugged into the cost model of DT.

Data Distillation and Cleaning DT is an instance of the data augmentation problem with some additional conditions on the group counts [103]. Moreover, data distillation [104] is particularly applicable in determining the group that a sampled tuple is associated with if such information is absent. Moreover, data cleaning is included in the source preparation process and its cost can be folded into the cost model.

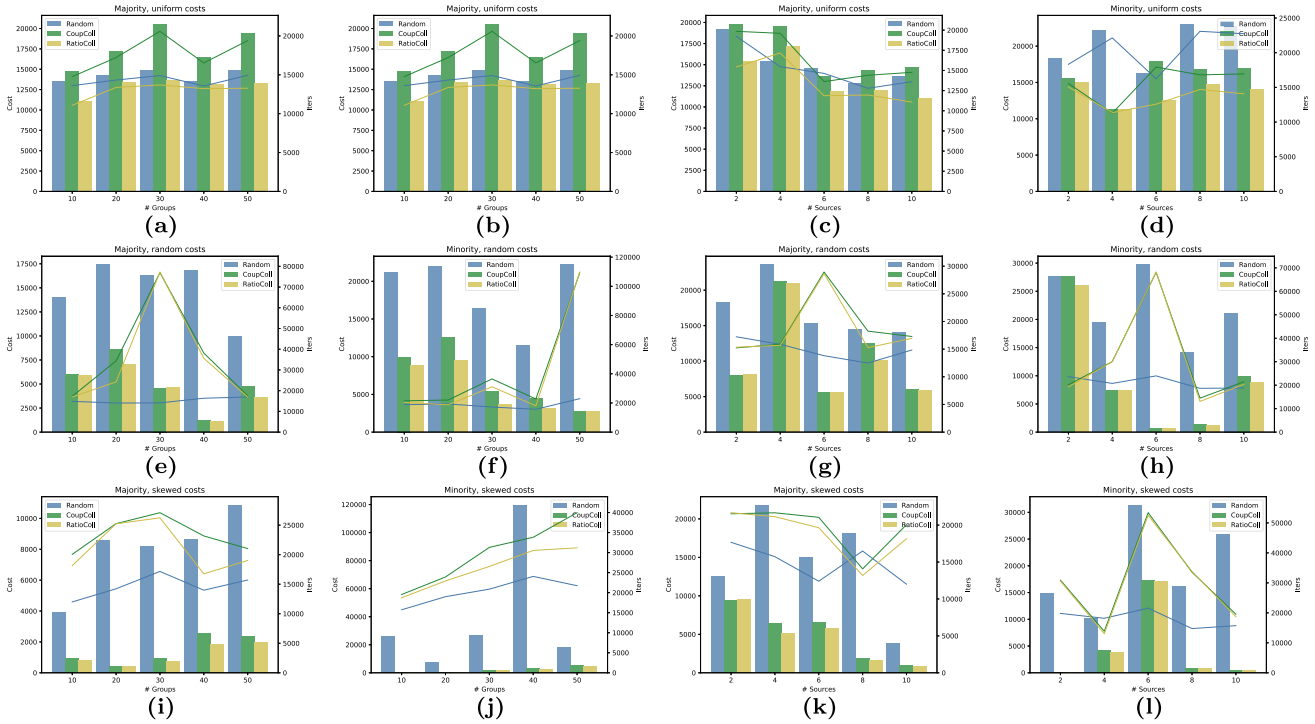


Fig. 5 Known DT for Minority and Majority Distributions and Equal, Random, and Skewed Cost Models

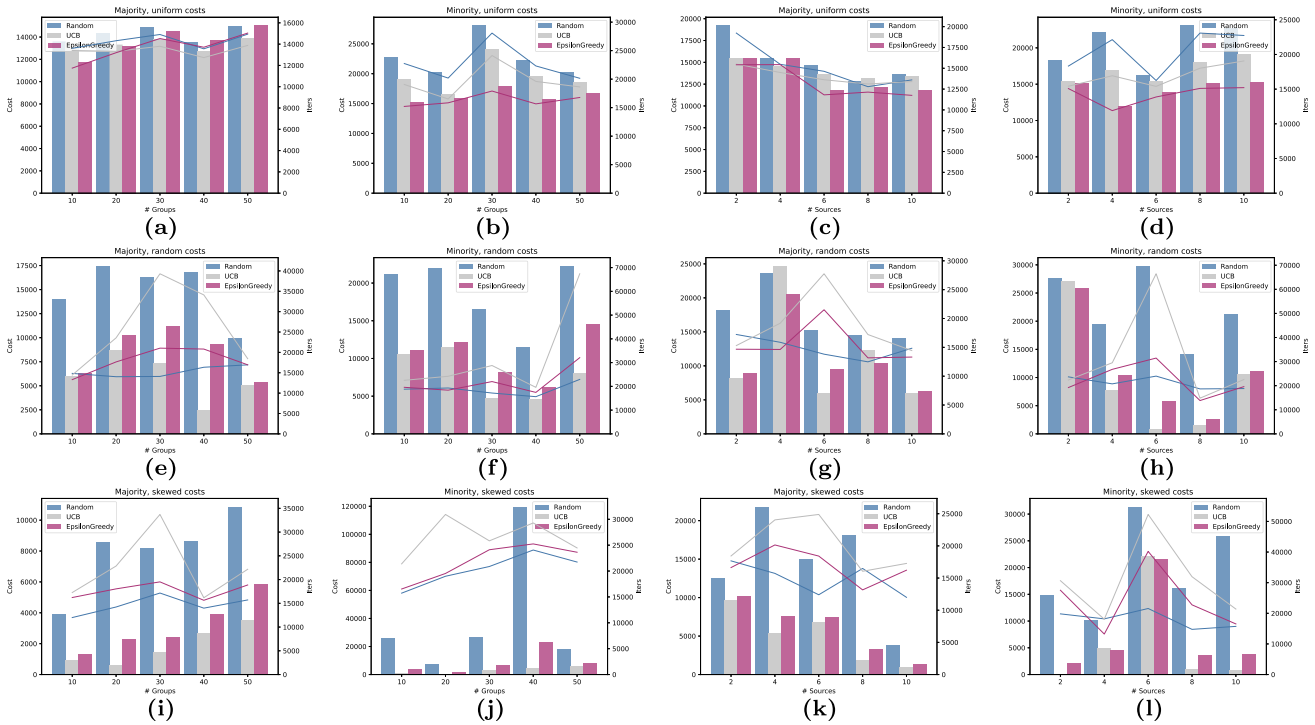


Fig. 6 Unknown DT for Minority and Majority Distributions and Equal, Random, and Skewed Cost Models

Cleaning tasks such as entity resolution are necessary for determining the freshness of samples.

Data Acquisition and Databases Matteo Brucato et al. extended SQL queries to handle package queries that specify complex constraints over answer sets [105, 106]. Package queries could be used to query representative data sets efficiently from an RDBMS. For instance, users could enforce

count constraints over labels and define a cost minimization objective.

7 Extensions

$k > 1$ Query Model So far in the paper, we assumed a data source returns one sample per query. First, if a query returns more than one tuple, all of those samples will be used to collect the target data set. In a setting where a query to a source returns more than one tuple ($k > 1$), typically, k is a small constant (e.g., 10). This will not require notable changes in the designed algorithms. For KNOWN- BINARY, except for the marginal cases, the algorithm remains near-optimal. Recall that KNOWN- BINARY keeps querying the source that has the highest ratio for the minorities. If the data source returns more than one sample, the algorithm still queries the same data source but it updates its counts using all returned tuples. This is equivalent to the algorithm calling the data source multiple times, something the optimal algorithm does, except in marginal cases where either the minority group changes or it finds a better data source. It is easy to see such marginal cases are unlikely to happen in practice. Even if it happens, such cases will reduce the cost by a small constant. The same argument is also valid for the COUPCOLL algorithm. We leave further investigations about these cases, as well as theoretical analyses of our algorithms under $k > 1$ query model, as part of our future work. The multi-armed bandit algorithms also work as-is for $k > 1$. The major impact of the new model on the algorithms is that, depending on the underlying distributions and the sizes $\mathcal{Q}_j$, the bandit algorithms may not have enough "time" to effectively identify the good data sources to query. As a result, its performance advantage compared to baseline may decline.

Count Requirements on Multiple Groups The count requirements may be on multiple groups individually, for example, we may need 100 of `gender`=F and 100 of `gender`=M as well as 100 of `race`=W and 100 of `race`=NW. We can achieve this target by performing a sequence of independent DTs for group requirements. We start by a DT that collects a target data set that satisfies the requirements of one group. In the following DT instances, tuples of the current target data set are replaced with new tuples of required groups while making sure that the counts of the groups of previous runs remain unchanged.

Complex Distributions on Groups We may have scenarios that require more sophisticated distribution functions on groups rather than count requirements. For example, a count requirement may be a range, i.e., as soon as the count of a group becomes equal to or greater than the lower bound of a range interval, the requirement is satisfied and the algorithm must start discarding samples of this group once the count becomes equal to the upper bound.

Overlapping Sources In real world, independent data sources have minimal overlap and we did not consider the overlap between sources in our optimization. For future work, we design algorithms that further optimize the cost, using the information about overlaps.

Diversity Maximization A potential downside of the cost minimization objective in DT is that it tends to sample heavily from a small number of most cost-efficient sources. The distribution of the unified data set could potentially be skewed compared to the ground truth, which is problematic in AI/ML applications where i.i.d. sampling is often assumed. A possible remedy is to incorporate diversity maximization as a constraint or objective. If a single data source only covers a small region of the entire vector space, maximizing a diversity metric such as max-min diversity [107] would entail sampling from a variety of data sources. We could adapt existing algorithms for diversity maximization under fairness constraint [108, 109] into the DP framework. A diversity-maximizing DP algorithm would avoid repeatedly sampling from data sources that cover highly overlapping regions of the vector space while staying under a cost budget.

8 Conclusions

In this paper, we studied the DT problem, which aims to cost-efficiently gather a unified data set from many data sources with fairness constraints. In the known statistics case, we developed the RATIOCOLL algorithm which has a tight asymptotic expected cost and empirically outperforms the previously proposed COUPCOLL algorithm. In the unknown statistics case, we generalized the problem to situations in which the heuristic proposed in [1] is not applicable. We developed the EPSILONGREEDY bandit algorithm that does not require any prior information. In practice, it performs on par with UCB, even though it uses less information.

Our practical recommendation is to use RATIOCOLL when probability distributions are known, and EXPLOREEXPLOIT otherwise. Furthermore, the duplicate-aware variants should be preferred for all but the most massive data sources.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00778-024-00849-w.

References

1. Nargesian, F., Asudeh, A., Jagadish, H.V.: Tailoring data source distributions for fairness-aware data integration. Proceed. VLDB Endow. **14**(11), 2519–2532 (2021). https://doi.org/10.14778/3476249.3476299
2. Rose, A.: Are face-detection cameras racist? Time Business (2010)

3. Mulshine, M.: A major flaw in google's algorithm allegedly tagged two black people's faces with the word 'gorillas'. Business Insider (2015)
4. Townsend, T.: Most engineers are white and so are the faces they use to train software. Recode (2017)
5. Dastin, J.: Amazon scraps secret ai recruiting tool that showed bias against women. Reuters (2018)
6. Holt, D., Elliot, D.: Methods of weighting for unit non-response. J. R. Stat. Soc. Series D (The Statistician) **40**(3), 333–342 (1991)
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
8. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsl **6**(1), 20–29 (2004)
9. Parsa, A.B., Taghipour, H., Derrible, S., Mohammadian, A.K.: Real-time accident detection: coping with imbalanced data. Accident Anal. Prevent. **129**, 202–210 (2019)
10. Chung, Y., Kraska, T., Polyzotis, N., Tae, K.H., Whang, S.E.: Slice finder: Automated data slicing for model validation. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 1550–1553. IEEE (2019)
11. Sagadeeva, S., Boehm, M.: Sliceline: Fast, linear-algebra-based slice finding for ml model debugging. In: Proceedings of the 2021 International Conference on Management of Data, pp. 2290–2299 (2021)
12. Tae, K.H., Whang, S.E.: Slice tuner: A selective data acquisition framework for accurate and fair machine learning models. In: Proceedings of the 2021 International Conference on Management of Data, pp. 1771–1783 (2021)
13. Bartlett, R., Morse, A., Stanton, R., Wallace, N.: Consumer-lending discrimination in the fintech era. Tech. rep, National Bureau of Economic Research (2019)
14. Dawex: Dawex: Sell, buy and share data. https://www.dawex.com/en
15. Xignite: Market data solutions. https://www.xignite.com/
16. WorldQuant: Worldquant. https://www.worldquant.com
17. Singer, N.: A data broker offers a peek behind the curtain. The New York Times (2013)
18. of California, S.: Data broker registry. https://oag.ca.gov/data-brokers (2020)
19. Turk, A.M.: Amazon mechanical turk. Retrieved August **17**, 2012 (2012)
20. Nargesian, F., Zhu, E., Pu, K.Q., Miller, R.J.: Table union search on open data. PVLDB **11**(7), 813–825 (2018)
21. Rapid: Google flights api: Incorporate travel data into your app. The Rapid API Blog (2020)
22. Chai, C., Fan, J., Li, G.: Incentive-based entity collection using crowdsourcing. In: ICDE, pp. 341–352 (2018)
23. Fan, J., Wei, Z., Zhang, D., Yang, J., Du, X.: Distribution-aware crowdsourced entity collection. IEEE Trans. Knowl. Data Eng. **31**(7), 1312–1326 (2019)
24. Chai, C., Li, G., Li, J., Deng, D., Feng, J.: Cost-effective crowdsourced entity resolution: a partial-order approach. In: SIGMOD, pp. 969–984 (2016)
25. Asudeh, A., Nargesian, F.: Towards distribution-aware query answering in data markets. Proc. VLDB Endow. **15**(11), 3137–3144 (2022)
26. The texas tribune data set. https://salaries.texastribune.org (2021)
27. Luo, G., Ellmann, C.J., Haas, P.J., Naughton, J.F.: A scalable hash ripple join algorithm. In: SIGMOD, pp. 252–262 (2002)
28. Li, F., Wu, B., Yi, K., Zhao, Z.: Wander join: online aggregation via random walks. In: SIGMOD, pp. 615–629 (2016)
29. Zhao, Z., Christensen, R., Li, F., Hu, X., Yi, K.: Random sampling over joins revisited. In: SIGMOD, pp. 1525–1539 (2018)
30. The socrata open data api. https://developer.twitter.com/en/products/twitter-api/enterprise
31. Li, Y., Yu, X., Koudas, N.: Data acquisition for improving machine learning models. Proc. VLDB Endow. **14**(10), 1832–1844 (2021)
32. Sheng, C., Zhang, N., Tao, Y., Jin, X.: Optimal algorithms for crawling a hidden database in the web. arXiv preprint arXiv:1208.0075 (2012)
33. Madhavan, J., Ko, D., Kot, Ł, Ganapathy, V., Rasmussen, A., Halevy, A.: Google's deep web crawl. Proceed. VLDB Endow. **1**(2), 1241–1252 (2008)
34. Asudeh, A., Thirumuruganathan, S., Zhang, N., Das, G.: Discovering the skyline of web databases. PVLDB **9**(7), 600–611 (2016)
35. Asudeh, A., Zhang, N., Das, G.: Query reranking as a service. PVLDB **9**(11), 888–899 (2016)
36. Sundarkumar, G.G., Ravi, V.: A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance. Eng. Appl. Artif. Intell. **37**, 368–377 (2015)
37. Select Issues: Assessing Adverse Impact in Software, Algorithms, and Artificial Intelligence Used in Employment Selection Procedures Under Title VII of the Civil Rights Act of 1964 (2023)
38. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 259–268 (2015)
39. Kearns, M., Neel, S., Roth, A., Wu, Z.S.: Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In: International conference on machine learning, pp. 2564–2572. PMLR (2018)
40. Kearns, M., Neel, S., Roth, A., Wu, Z.S.: An empirical study of rich subgroup fairness for machine learning. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 100–109 (2019)
41. Foulds, J.R., Islam, R., Keya, K.N., Pan, S.: An intersectional definition of fairness. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 1918–1921. IEEE (2020)
42. Asudeh, A., Jin, Z., Jagadish, H.V.: Assessing and remedying coverage for a given dataset. In: ICDE, pp. 554–565 (2019)
43. Deng, S., Lu, S., Tao, Y.: On join sampling and the hardness of combinatorial output-sensitive join algorithms. In: PODS, pp. 99–111. ACM (2023)
44. Asudeh, A., Nargesian, F.: Towards distribution-aware query answering in data markets. Proc. VLDB Endow. **15**(11), 3137–3144 (2022)
45. Bird, R.S.: Tabulation techniques for recursive programs. ACM Comput. Surveys **12**(4), 403–417 (1980). https://doi.org/10.1145/356827.356831
46. Aggarwal, A., Klawe, M.M., Moran, S., Shor, P., WIlber, R.: Geometric applications of a matrix searching algorithm. In: Proceedings of the Second Annual Symposium on Computational Geometry (1986). https://dl.acm.org/doi/pdf/10.1145/10515.10546
47. Galil, Z., Park, K.: Dynamic programming with convexity, concavity and sparsity. Theor. Comput. Sci. **92**(1), 49–76 (1992). https://doi.org/10.1016/0304-3975(92)90135-3
48. Motwani, R., Raghavan, P.: Randomized algorithms. Cambridge university press (1995)
49. Brown, M., Ross, S.M.: Optimality results for coupon collection. J. Appl. Probab. **53**(3), 930–937 (2016)
50. Katehakis, M.N., Jr., A.F.V.: The multi-armed bandit problem: Decomposition and computation. Math. Oper. Res. **12**(2), 262–268 (1987)
51. Bubeck, S., Cesa-Bianchi, N.: Regret analysis of stochastic and nonstochastic multi-armed bandit problems. Found. Trends Mach. Learn. **5**(1), 1–122 (2012)

52. Slivkins, A.: Introduction to Multi-Armed Bandits. Foundations and Trends® in Machine Learning **12**(1-2), 1–286 (2019). https://doi.org/10.1561/2200000068
53. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**, 235–256 (2002)
54. of Transportation Statistics, B.: Airborne flights database. U.S. Department of Transportation, https://www.transtats.bts.gov (2021)
55. ProPublica: Compas-analysis. ProPublica (2023). https://github.com/propublica/compas-analysis
56. Mattu, J., Angwin, L., Kirchner, S., Larson, J.: How We Analyzed the COMPAS Recidivism Algorithm (2016). https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm?token=TiqCeZIj4uLbXl91e3wM2PnmnWbCVOvS
57. Lagioia, F., Rovatti, R., Sartor, G.: Algorithmic fairness through group parities? the case of compas-sapmoc. AI & SOCIETY pp. 1–20 (2022)
58. Fabris, A., Messina, S., Silvello, G., Susto, G.A.: Algorithmic fairness datasets: the story so far. Data Min. Knowl. Disc. **36**(6), 2074–2152 (2022)
59. Barocas, S., Hardt, M., Narayanan, A.: Fairness and machine learning: Limitations and opportunities. URL: fairmlbook.org (2019)
60. Friedler, S.A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E.P., Roth, D.: A comparative study of fairness-enhancing interventions in machine learning. In: Proceedings of the conference on fairness, accountability, and transparency, pp. 329–338 (2019)
61. Kamiran, F., Calders, T.: Data preprocessing techniques for classification without discrimination. Knowl. Inf. Syst. **33**(1), 1–33 (2012)
62. Calmon, F., Wei, D., Vinzamuri, B., Ramamurthy, K.N., Varshney, K.R.: Optimized pre-processing for discrimination prevention. In: Advances in Neural Information Processing Systems, pp. 3992–4001 (2017)
63. Salimi, B., Rodriguez, L., Howe, B., Suciu, D.: Interventional fairness: Causal database repair for algorithmic fairness. In: SIGMOD, pp. 793–810 (2019)
64. Kamishima, T., Akaho, S., Asoh, H., Sakuma, J.: Fairness-aware classifier with prejudice remover regularizer. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 35–50. Springer (2012)
65. Zemel, R., Wu, Y., Swersky, K., Pitassi, T., Dwork, C.: Learning fair representations. In: ICML (2013)
66. Zafar, M.B., Valera, I., Rodriguez, M.G., Gummadi, K.P.: Fairness constraints: Mechanisms for fair classification. CoRR, abs/1507.05259 (2015)
67. Zhang, H., Chu, X., Asudeh, A., Navathe, S.: Omnifair: A declarative system for model-agnostic group fairness in machine learning. SIGMOD (2021)
68. Kamiran, F., Calders, T., Pechenizkiy, M.: Discrimination aware decision tree learning. In: 2010 IEEE International Conference on Data Mining, pp. 869–874. IEEE (2010)
69. Hardt, M., Price, E., Srebro, N.: Equality of opportunity in supervised learning. arXiv preprint arXiv:1610.02413 (2016)
70. Woodworth, B., Gunasekar, S., Ohannessian, M.I., Srebro, N.: Learning non-discriminatory predictors. In: Conference on Learning Theory, pp. 1920–1953. PMLR (2017)
71. Salimi, B., Howe, B., Suciu, D.: Database repair meets algorithmic fairness. ACM SIGMOD Rec. **49**(1), 34–41 (2020)
72. Asudeh, A., Jagadish, H., Stoyanovich, J., Das, G.: Designing fair ranking schemes. In: SIGMOD, pp. 1259–1276 (2019)
73. Kuhlman, C., Rundensteiner, E.: Rank aggregation algorithms for fair consensus. PVLDB **13**(12), 2706–2719 (2020)
74. Asudeh, A., Jagadish, H., Miklau, G., Stoyanovich, J.: On obtaining stable rankings. PVLDB **12**(3) (2019)
75. Guan, Y., Asudeh, A., Mayuram, P., Jagadish, H., Stoyanovich, J., Miklau, G., Das, G.: Mithraranking: A system for responsible ranking design. In: SIGMOD, pp. 1913–1916 (2019)
76. Sun, C., Asudeh, A., Jagadish, H., Howe, B., Stoyanovich, J.: Mithralabel: Flexible dataset nutritional labels for responsible data science. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2893–2896 (2019)
77. Yang, K., Stoyanovich, J., Asudeh, A., Howe, B., Jagadish, H., Miklau, G.: A nutritional label for rankings. In: SIGMOD, pp. 1773–1776 (2018)
78. Getoor, L.: Responsible data science. In: SIGMOD (2019)
79. Stoyanovich, J., Howe, B., Jagadish, H.: Responsible data management. PVLDB **13**(12), 3474–3488 (2020)
80. Shah, N.B., Lipton, Z.: Sigmod 2020 tutorial on fairness and bias in peer review and other sociotechnical intelligent systems. In: SIGMOD, pp. 2637–2640 (2020)
81. Venkatasubramanian, S.: Algorithmic fairness: measures, methods and representations. In: PODS, pp. 481–481 (2019)
82. Asudeh, A., Jagadish, H.V.: Fairly evaluating and scoring items in a data set. PVLDB **13**(12), 3445–3448 (2020)
83. Neyman, J., Pearson, E.S.: Contributions to the theory of testing statistical hypotheses. Stat. Res. Memoirs (1936)
84. Olteanu, A., Castillo, C., Diaz, F., Kiciman, E.: Social data: Biases, methodological pitfalls, and ethical boundaries. Front. Big Data **2**, 13 (2019)
85. Barocas, S., Selbst, A.D.: Big data's disparate impact. Calif. L. Rev. **104**, 671 (2016)
86. Chen, I., Johansson, F.D., Sontag, D.: Why is my classifier discriminatory? In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (eds.) Advances in Neural Information Processing Systems, vol. 31, pp. 3539–3550 (2018)
87. Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., Wallach, H.: Improving fairness in machine learning systems: What do industry practitioners need? In: Proceedings of the 2019 CHI conference on human factors in computing systems, pp. 1–16 (2019)
88. Drosou, M., Jagadish, H., Pitoura, E., Stoyanovich, J.: Diversity in big data: A review. Big data **5**(2) (2017)
89. Lin, Y., Guan, Y., Asudeh, A., V., J.H.: Identifying insufficient data coverage in databases with multiple relations. PVLDB **13**(11), 2229–2242 (2020)
90. Jin, Z., Xu, M., Sun, C., Asudeh, A., Jagadish, H.: Mithracoverage: A system for investigating population bias for intersectional fairness. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 2721–2724 (2020)
91. Accinelli, C., Minisi, S., Catania, B.: Coverage-based rewriting for data preparation. In: EDBT/ICDT Workshops (2020)
92. Asudeh, A., Shahbazi, N., Jin, Z., Jagadish, H.: Identifying insufficient data coverage for ordinal continuous-valued attributes. SIGMOD (2021)
93. Orr, L.J., Balazinska, M., Suciu, D.: Sample debiasing in the themis open world database system. In: SIGMOD, pp. 257–268 (2020)
94. Zhu, E., Nargesian, F., Pu, K.Q., Miller, R.J.: LSH ensemble: internet-scale domain search. PVLDB **9**(12), 1185–1196 (2016)
95. Sadiq, S.W., Dasu, T., Dong, X.L., Freire, J., Ilyas, I.F., Link, S., Miller, R.J., Naumann, F., Zhou, X., Srivastava, D.: Data quality: The role of empiricism. SIGMOD Rec. **46**(4), 35–43 (2017)
96. Rekatsinas, T., Deshpande, A., Dong, X.L., Getoor, L., Srivastava, D.: Sourcesight: Enabling effective source selection. In: SIGMOD, pp. 2157–2160 (2016)

97. Shen, Y., Chakrabarti, K., Chaudhuri, S., Ding, B., Novik, L.: Discovering queries based on example tuples. In: SIGMOD, pp. 493–504 (2014)
98. Qian, L., Cafarella, M.J., Jagadish, H.V.: Sample-driven schema mapping. In: SIGMOD, pp. 73–84 (2012)
99. Lehmberg, O., Bizer, C.: Synthesizing n-ary relations from web tables. In: WIMS, pp. 17:1–17:12 (2019)
100. Pimplikar, R., Sarawagi, S.: Answering table queries on the web using column keywords. PVLDB **5**(10), 908–919 (2012)
101. Brickley, D., Burgess, M., Noy, N.F.: Google dataset search: Building a search engine for datasets in an open web ecosystem. In: WWW, pp. 1365–1375 (2019)
102. Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., Suciu, D.: Query-based data pricing. J. ACM **62**(5), 43:1–43:44 (2015)
103. Chepurko, N., Marcus, R., Zgraggen, E., Fernandez, R.C., Kraska, T., Karger, D.: ARDA: automatic relational data augmentation for machine learning. PVLDB **13**(9), 1373–1387 (2020)
104. Radosavovic, I., Dollár, P., Girshick, R.B., Gkioxari, G., He, K.: Data distillation: Towards omni-supervised learning. In: CVPR, pp. 4119–4128 (2018)
105. Brucato, M., Beltran, J.F., Abouzied, A., Meliou, A.: Scalable package queries in relational database systems. arXiv preprint arXiv:1512.03564 (2015)
106. Brucato, M., Mannino, M., Abouzied, A., Haas, P.J., Meliou, A.: spaqltools: a stochastic package query interface for scalable constrained optimization. Proceedings of the VLDB Endowment **13**(12) (2020)
107. Erkut, E.: The discrete p-dispersion problem. Eur. J. Oper. Res. **46**(1), 48–60 (1990)
108. Wang, Y., Fabbri, F., Mathioudakis, M.: Streaming algorithms for diversity maximization with fairness constraints. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 41–53. IEEE (2022)
109. Wang, Y., Mathioudakis, M., Li, J., Fabbri, F.: Max-min diversification with fairness constraints: Exact and approximation algorithms. In: SIAM nternational Conference on Data Mining (SDM23) (2023)