Cost-efficient Data Acquisition on Online Data Marketplaces for Correlation Analysis

Yanying Li, Haipei Sun
Stevens Institute of Technology
1 Castle Point Terrace
Hoboken, New Jersey 07030
yli158,hsun15@stevens.edu

Boxiang Dong
Montclair State University
1 Normal Ave
Montclair, New Jersey 07043
dongb@montclair.edu

Hui (Wendy) Wang
Stevens Institute of Technology
1 Castle Point Terrace
Hoboken, New Jersey 07030
Hui.Wang@stevens.edu

ABSTRACT

Incentivized by the enormous economic profits, the data marketplace platform has been proliferated recently. In this paper, we consider the data marketplace setting where a data shopper would like to buy data instances from the data marketplace for correlation analysis of certain attributes. We assume that the data in the marketplace is dirty and not free. The goal is to find the data instances from a large number of datasets in the marketplace whose join result not only is of high-quality and rich join informativeness, but also delivers the best correlation between the requested attributes. To achieve this goal, we design DANCE, a middleware that provides the desired data acquisition service. DANCE consists of two phases: (1) In the *off-line* phase, it constructs a two-layer join graph from samples. The join graph includes the information of the datasets in the marketplace at both schema and instance levels; (2) In the *on-line* phase, it searches for the data instances that satisfy the constraints of data quality, budget, and join informativeness, while maximizing the correlation of source and target attribute sets. We prove that the complexity of the search problem is NP-hard, and design a heuristic algorithm based on Markov chain Monte Carlo (MCMC). Experiment results on two benchmark and one real datasets demonstrate the efficiency and effectiveness of our heuristic data acquisition algorithm.

PVLDB Reference Format:
Yanying Li, Haipei Sun, Boxiang Dong, Wendy Hui Wang. Cost-efficient Data Acquisition on Online Data Marketplaces for Correlation Analysis. *PVLDB*, 12(4): 362-375, 2018.
DOI: https://doi.org/10.14778/3297753.3297757

1. INTRODUCTION

With the explosion in the profits from analyzing the data, data has been recognized as a valuable commodity. Recent research [4] predicts that the sales of big data and analytics will reach $187 billion by 2019. Incentivized by the enormous economic profits, the *data marketplace* model was proposed recently [6]. In this model, data is considered as asset for purchase and sale. Data marketplaces in the cloud enable to sell the (cheaper) data by providing Web platforms for buying and selling data; examples include Microsoft Azure Marketplace [3] and BDEX [1].

In this paper, we consider the data shopper who has a particular type of data analytics needs as *correlation analysis of some certain attributes*. He may own a subset of these attributes already. He intends to purchase more attributes from the data marketplace to perform the correlation analysis. However, many existing cloud-based data marketplaces (e.g., Microsoft Azure Marketplace [3] and Google Big Query service [2]) do not support efficient exploration of datasets of interest. The data shopper has to identify the relevant data instances through a non-trivial data search process. First, he identifies which datasets are potentially useful for correlation analysis based on the (brief) description of the datasets provided by the data marketplaces, which normally stays at schema level. There may exist multiple datasets that appear relevant. This leads to multiple purchase options. Based on these options, second, the shopper has to decide which purchase option is the best (i.e., it returns the maximum correlation of the requested attributes) under his constraints (e.g., data purchase budget). Let us consider the following example for more details.

Example 1.1. *Consider the data shopper Adam, a data research scientist, who has a hypothesis regarding the correlation between age groups and diseases in New Jersey (NJ), USA. Adam only has the information of* age, zipcode, *and* population *in his own dataset D_S (Figure 1 (a)). To test his hypothesis, Adam plans to purchase additional data from the marketplace. Assume that he identifies five instances D_1 - D_5 (Figure 1 (b)-(f)) in the data marketplace as relevant. There are several data purchase options, with four of them shown below:*

- *Option 1: Purchase D_1 and D_2;*
- *Option 2: Purchase three attributes (Gender, Disease, # of cases) of D_3 and (Age, Gender, Population) of D_4;*
- *Option 3: Purchase three attributes (Race, Disease, # of cases) of D_3 and (Age, Race, Population) of D_4;*
- *Option 4: Purchase D_5;*

For each option, the correlation analysis is performed on the join result of the purchased data and the shopper's local one. In general, the number of possible join paths is exponential to the number of datasets in the marketplace. The brute-force method that enumerates all join paths to

Proceedings of the VLDB Endowment, Vol. 12, No. 4
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3297753.3297757

Age	Zipcode	Population
[35, 40]	10003	7,000
[20, 25]	01002	3,500
[55, 60]	07003	1,200
[35, 40]	07003	5,800
[35, 40]	07304	2,000

(a) D_S: *Source* instance owned by data shopper Adam

Zipcode	State	
07003	NJ	correct
07304	NJ	correct
10001	NY	correct
10001	NJ	wrong

(b) D_1: Zipcode table

State	Disease	# of cases
MA	Flu	300
NJ	Flu	400
Florida	Lyme disease	130
California	Lyme disease	40
NJ	Lyme disease	200

(c) D_2: Data and statistics of diseases by state

Gender	Race	Disease	# of cases
M	White	Flu	200
F	Asian	AIDS	30
M	White	Diabetes	4,000
M	Hispanic	Flu	140

(d) D_3: Data and statistics of diseases of New Jersey (NJ) State by gender

Age	Gender	Race	Population
[35,40]	M	White	400,000
[20,25]	F	Asian	100,000
[20,25]	M	White	300,000
[40,45]	M	Hispanic	50,000

(e) D_4: Census dataset of New Jersey (NJ) State

Age	Address	Insurance	Disease
[35, 40]	10 North St.	UnitedHealthCare	Flu
[20, 25]	5 Main St.	MedLife	HIV
[35, 40]	25 South St.	UnitedHealthCare	Flu

(f) D_5: Insurance & disease data instance

Figure 1: An example of data acquisition

find the one that delivers the best correlation between source and target attributes is prohibitively expensive. Besides the scalability issue, there are several important issues for data purchase on the marketplace. Next, we discuss them briefly.

Meaningful joins. Different joins return different amounts of useful information. For example, consider Option 2 and 3 in Example 1.1. The join result in Option 2 is grouped by gender, while that of Option 3 is grouped by race. Both join results are considered as meaningful, depending on how the shopper will test his hypothesis on the join results. On the other hand, the join result of Option 4 is meaningless, as it associates the aggregation data with individual records.

Data quality. The data on the data marketplaces tends to be dirty [14]. In this paper, we consider *data consistency* as the main data quality issue. Intuitively, data quality (in terms of its consistency) is measured as the percentage of data content that is consistent with the given *integrity constraints* which take the form of functional dependency (FD) [11, 8]. Consider D_1 in Table 1 (b) that has a FD: *Zipcode* → *State* (i.e., the same Zipcode values are associated with the same State value). The last record is inconsistent with the FD. Thus Option 1 in Example 1.1 can lead to errors in the hypothesis on the join results. A naive solution is to clean the data in the marketplace off-line before data purchase. However, we will show that join indeed can impact the data quality significantly: the join of high-quality (low-quality, resp.) data instances can become low-quality (high-quality, resp.). Therefore, the data quality issue has to be dealt online during data acquisition.

Constrained budget. We assume the data shopper is equipped with a limited budget for data purchase. We follow the query-based pricing model [6] used by some commercial cloud-based data marketplaces (e.g., Google Big Query service [2]): a data shopper submits SQL queries to the data marketplaces and pays for the query results. This pricing model allows the data shopper to only purchase the necessary attributes instead of the whole dataset. However, choosing which attributes to purchase under budget constraint introduces additional complexity to data purchase.

Data scalability, data quality, join informativeness, and budget constraints make data purchase from the data marketplaces extremely challenging for the data shoppers, especially for those common users who do not have much experience and expertise in data management. The existing works on data exploration [37, 28, 27] mainly focus on finding the best join paths among multiple datasets in terms of join size and/or informativeness. However, these solutions cannot be easily adapted to data marketplaces, since it is too expensive to construct the schema graph [28] for all the data on the marketplace due to its large scale. Furthermore, the search criteria for join path is different. While these existing works on data exploration try to find the join path of the best informativeness, our work aims to find the best join path that maximizes the correlation of a set of specific attributes, with the presence of the constraints such as data quality and budgets.

Contributions. We design a middleware service named DANCE, a Data Acquisition framework on oNline data market for CorrElation analysis, that provides cost-efficient data acquisition service for the budget-conscious search of the high-quality data on the data marketplaces, aiming to maximize the correlation of certain attributes. DANCE consists of two phases: (1) during the *off-line* phase, it constructs a join graph of the datasets on the data marketplace. The join graph contains the information of these datasets at both schema and instance levels. To deal with the data scalability issue of data acquisition, DANCE collects samples from the data marketplace, and constructs the join graph from the samples; (2) during the *online* phase, it accepts the data shoppers' correlation request as well as the constraints (budgets, data quality, and join informativeness). By leveraging the join graph, it estimates the correlation of a certain set of attributes, as well as the join informativeness and quality of the data items to be purchased. Based on these estimated information, DANCE recommends the data items on the data marketplace that deliver the best correlation while satisfying the given constraints to the data shopper.

We make the following contributions. First, we design a new sampling method named *correlated re-sampling* to deal with multi-table joins of large intermediate join size. We show how to unbiasedly estimate the correlation, join informativeness, and quality based on the samples. Second, we design a new two-layer *join graph* structure. The join graph is constructed from the samples collected from the data marketplace. It includes the join relationship of the data instances at schema level, as well as the information of correlation, join informativeness, quality, and price of the data at instance level. Based on the join graph, third, we transform the data acquisition problem to a graph search problem, which searches for the optimal subgraph in the join graph that maximizes the (estimated) correlation between source and target nodes, while the quality, join informativeness (in format of edge weights in the graph), and price satisfy the specified constraints. We prove that the problem of searching for such optimal subgraph is NP-hard. Thus, we design a heuristic algorithm based on Markov chain Monte

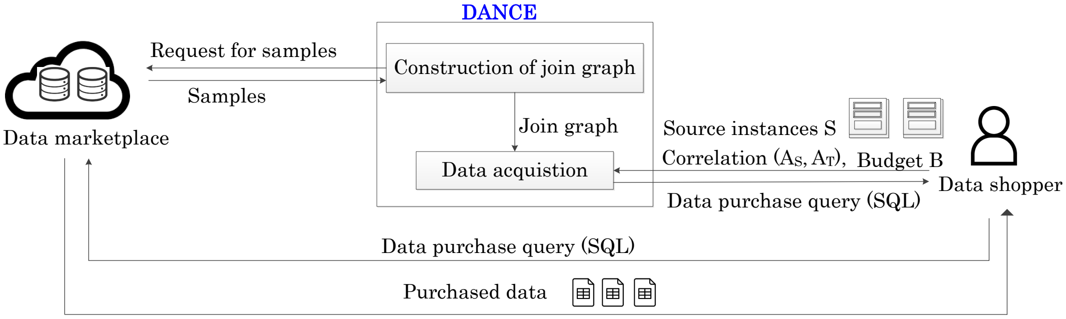


Figure 2: Framework of DANCE

Carlo (MCMC). Our heuristic algorithm first searches for the minimal weighted graph at the instance layer of the join graph, which corresponds to a set of specific instances. Then it finds the optimal target graph at the attribute set layer of the join graph, which corresponds to the attributes in those instances that are to be purchased from the marketplace. Last but not least, we perform an extensive set of experiments on large-scale datasets. The results show that our search method can find the acquisition results with high correlation efficiently.

The rest of the paper is organized as following. Section 2 introduces the preliminaries. Section 3 presents the data sampling method. Section 4 and Section 5 present the off-line and online phases of DANCE respectively. Section 6 shows the experiment results. Section 7 discusses possible extensions and future work. Section 8 discusses the related work. Section 9 concludes the paper.

2. PRELIMINARIES

2.1 System Overview

We consider a data marketplace **M** that stores and maintains a collection of relational database instances $\mathcal{D} = \{D_1, \ldots, D_n\}$. A *data shopper* may own a set of relational data instances $\mathbb{S}$ locally, which contain a set of *source attributes* $\mathcal{A}_{\mathcal{S}}$. The data shopper would like to purchase a set of *target attributes* $\mathcal{A}_{\mathcal{T}}$ from **M**, so that the correlation between $\mathcal{A}_{\mathcal{S}}$ and $\mathcal{A}_{\mathcal{T}}$ is maximized. In this paper, we only consider *vertical purchase*, i.e., the shopper buys the whole set of data values of certain attributes from the data marketplaces. We assume the data shoppers fully trust DANCE. Figure 2 illustrates the framework of DANCE. It consists of two phases: *off-line* and *online* phases.

Off-line phase. DANCE collects a set of samples from **M**, and constructs a data structure named *join graph* that includes the join relationship among the samples at schema level, and the join informativeness at instance level.

Online phase. DANCE accepts the acquisition requests from data shoppers. Each acquisition request consists of: (1) the source dataset $\mathbb{S}$ that is owned by the shopper and the source attributes $\mathcal{A}_{\mathcal{S}} \in \mathbb{S}$; (2) the target attributes $\mathcal{A}_{\mathcal{T}} \in \mathcal{D}$, and (3) the budget B of data purchase from **M**, to DANCE. The data shopper can also specify his requirement of data quality and join informativeness by the setup of the corresponding threshold values. Both $\mathbb{S}$ and $\mathcal{A}_{\mathcal{S}}$ are optional. The acquisition request without $\mathbb{S}$ and $\mathcal{A}_{\mathcal{S}}$ aims to find the best correlation of $\mathcal{A}_{\mathcal{T}}$ in $\mathcal{D}$.

After DANCE receives the acquisition request, it first processes the request on its local join graph by searching for a set of *target instances* $\mathbb{T} = \{T_1, \ldots, T_k\}$, where: (1) for each T_i, there exists an instance $D_j \in \mathcal{D}$ such that $T_i \subseteq D_j$; (2) the correlation between $\mathcal{A}_{\mathcal{S}}$ and $\mathcal{A}_{\mathcal{T}}$ is maximized in $\mathcal{J} = \bowtie_{\forall T_i \in \mathbb{S} \cup \mathbb{T}} T_i$, where $\bowtie$ denotes the equi-join operator; (3) each $T_i \in \mathbb{T}$ is assigned a *price*, which is decided by a query-based pricing function [6]. The total price of $\mathbb{T}$ should not exceed B; and (4) the quality and join informativeness of $\mathcal{J}$ satisfy the thresholds specified by the data shopper. If no such target instance can be identified from its current join graph and data samples, DANCE purchases more samples from **M**, updates its local join graph, and performs the data search again. The iterative process continues until either the desired $\mathbb{T}$ is found or the data shopper changes his acquisition requirement (e.g., by relaxing the data quality threshold).

After DANCE identifies the target instances $\mathbb{T}$, it generates a set of SQL projection queries $\mathcal{Q}$, where each instance $T_i \in \mathbb{T}$ corresponds to a query $\mathsf{Q} \in \mathcal{Q}$. Each $\mathsf{Q} \in \mathcal{Q}$, denoted as $\pi_{A_i}(D_i)$ (π : the projection operator), selects a set of attributes A_i from the instance D_i. We call the attribute set A_i the *projection* attribute set. In the paper, we use $D.A$ to specify the values of attribute A of the instance D.

Upon receiving the purchase option $\mathcal{Q}$ from DANCE, the client sends $\mathcal{Q}$ to **M** directly for data purchase, and obtains the corresponding instances.

2.2 Correlation Measurement

Quite a few correlation measurement functions (e.g. *Pearson correlation coefficient* , *token inverted correlation* [31], and *cross correlation (autocorrelation)* [25]) can evaluate the correlation of multiple attributes. However, they only can be used to deal with either categorical or numerical data, but not both. Therefore, we use the entropy based correlation measure [26] to quantify the correlation as it deals with both categorical and numerical data. We assume the readers' familiarity with information theory concepts.

Definition 2.1. **[Correlation [26]]** *Given a dataset D and two attribute sets X and Y, the* correlation *of X and Y $CORR(X, Y)$ is measured as*

- $CORR(X, Y) = H(X) - H(X|Y)$ *if X is categorical,*
- $CORR(X, Y) = h(X) - h(X|Y)$ *if X is numerical,*

where $H(X)$ is the Shannon entropy of X in D, and $H(X|Y)$ is the conditional entropy:

$$H(X|Y) = \sum H(X|y)p(y),$$

Furthermore, $h(X)$ is the cumulative entropy of attribute X

$$h(X) = -\int P(X \leq x) log P(X \leq x) dx,$$

and

$$h(X|Y) = -\int h(X|y)p(y)dy.$$

Intuitively, the entropy $H(X)/h(X)$ measures the uncertainty of values in attribute X, and the conditional entropy $H(X|Y)/h(X|Y)$ quantifies the uncertainty of X given the knowledge of Y. The correlation $CORR(X,Y)$ computes the reduction in uncertainty of X given the knowledge of Y. If X and Y are highly correlated, $CORR(X,Y)$ is large.

2.3 Join Informativeness

Quantifying how informative the join results are is challenging. Evaluating join informativeness simply by join size is not suitable. For example, consider Option 5 in Example 1.1 (i.e. join of D_S and D_5 in Table 1). The join is meaningless since it associates the aggregation data with individual records. But its join size is larger than the other options (e.g., join D_S with D_1) that are more meaningful. Therefore, we follow the join informativeness measurement in [37], which measures the join informativeness by using the well-known concepts from information theory. Briefly speaking, given a joint distribution (X,Y), let $I(X,Y)$ and $H(X,Y)$ denote the mutual information and entropy respectively. Then $D(X,Y) = \frac{H(X,Y)-I(X,Y)}{H(X,Y)} = 1 - \frac{I(X,Y)}{H(X,Y)}$ is a distance function [20]. This distance function has several nice properties (e.g., non-negative and symmetric) that also hold on the join results. The join informativeness is formally defined below.

Definition 2.2. **[Join Informativeness [37]]** *Given two instances D_1 and D_2, let J be their join attribute(s). The join informativeness of D_1 and D_2 is defined as*

$$JI(D_1, D_2) = 1 - \frac{I(D_1.J, D_2.J)}{H(D_1.J, D_2.J)}, \tag{1}$$

The join informativeness value always falls in the range [0, 1]. It is worth noting that the smaller $JI(D_1, D_2, J)$ is, the more informative is the join connection between D_1 and D_2.

2.4 Data Quality

In this paper, we consider the setting where data on the marketplaces is *dirty*. In this project, we mainly consider one specific type of dirty data, the *inconsistent* data, i.e., the data items that violate integrity constraints [5, 29]. A large number of existing works (e.g., [11, 8]) specify the data consistency in the format of *functional dependency (FD)*. Formally, given a relational dataset D, a set of attributes Y of D is said to be *functionally dependent* on another set of attributes X of D (denoted $X \rightarrow Y$) if and only if for any pair of records r_1, r_2 in D, if $r_1[X] = r_2[X]$, then $r_1[Y] = r_2[Y]$. For any FD $F : X \rightarrow Y$ where Y contains multiple attributes, F can be decomposed in to multiple FD rules $F' : X \rightarrow Y'$, with Y' containing a single attribute of Y. Thus for the following discussions, we only consider FDs that contain a single attribute at the right hand side.

Based on FDs, the data inconsistency can be measured as the amounts of data that does not satisfy the FDs. Before we formally define the measurement of data inconsistency, first, we formally define *partitions*. We use $r.A$ to denote the value of attribute A of record r.

Definition 2.3. **[Equivalence Class (EC) and Partitions]** *[15] The* equivalence class *(EC) of a tuple r with respect to an attribute set X, denoted as eq^r_X, is defined as $eq^r_X = \{r' | r.A = r'.A, \forall A \in X, \forall r' \in D\}$. The set $\pi_X = \{eq^r_X | r \in D\}$ is defined as a* partition *of D of the attribute set X. Informally, π_X is a collection of disjoint sets (ECs) of tuples, such that each set has a unique representative value of the set of attributes X, and the union of the sets equals to D.*

Given a dataset D and an FD $F : X \rightarrow Y$, where X and Y are a set of attributes, the degree of data inconsistency by F can be measured as the fraction of records that do not satisfy this FD. Following this, we define the data quality function $Q(D, F)$ for a given FD F on an instance D.

Definition 2.4. **[Data quality of one instance w.r.t. one FD]** *Given a data instance D and a FD $F : X \rightarrow Y$ on D, for any equivalence class $eq_x \in \pi_X$, the* correct equivalence class *in $\pi_{X \cup Y}$ is*

$$\begin{aligned} C(D, X \rightarrow Y, eq_x) = \{&eq_{xy} | eq_{xy} \in \pi_{X \cup Y}, eq_{xy} \subseteq eq_x, \\ &\nexists eq'_{xy} \in \pi_{X \cup Y} \ s.t. \ eq'_{xy} \subseteq eq_x \ and \\ &|eq'_{xy}| > |eq_{xy}|\}. \end{aligned} \tag{2}$$

In other words, $C(D, X \rightarrow Y, eq_x)$ is the largest equivalence class in $\pi_{X \cup Y}$ with the same value on X attributes as eq_x. If there are multiple such equivalence classes of the same size, we break the tie randomly.

Based on the definition of correct equivalence class, the set of correct records in D w.r.t. F is defined as:

$$C(D, X \rightarrow Y) = \bigcup_{eq_x \in \pi_X} C(D, X \rightarrow Y, eq_x). \tag{3}$$

The quality of a given data instance D w.r.t. $F : X \rightarrow Y$ is measured as:

$$Q(D, X \rightarrow Y) = \frac{|C(D, X \rightarrow Y)|}{|D|}.$$

Example 2.1. *Consider the data instance D_1 in Table 1 (b) and the FD $F : Zipcode \rightarrow State$. We refer the four tuples in D_1 as t_1 -t_4 by following their order in the table. The partition $\pi_{Zipcode}$ includes three equivalence classes, $eq^1_Z = \{t_1\}$, $eq^2_Z = \{t_2\}$ and $eq^3_Z = \{t_3, t_4\}$. The partition $\pi_{Zipcode \cup State}$ consists of four equivalence classes, i.e., $eq^1_{ZS} = \{t_1\}$, $eq^2_{ZS} = \{t_2\}$, $eq^3_{ZS} = \{t_3\}$, and $eq^4_{ZS} = \{t_4\}$. Thus, we have $Q(D_1, F) = 0.75$. Note that eq^4_{ZS} is considered as an error.*

Based on the definition of data quality of one instance w.r.t to one FD, now we are ready to define data quality of several joinable instances w.r.t to a set of FDs.

Definition 2.5. **[Data quality]** *Given a set of instances $\mathcal{D}$, let $\mathcal{J} = \bowtie_{\forall D_i \in \mathcal{D}} D_i$, and $\mathcal{F}$ be the set of FDs that hold on $\mathcal{J}$. The set of correct records of D w.r.t. $\mathcal{F}$ is defined as $C(\mathcal{J}, \mathcal{F}) = \bigcap_{\forall F_i \in \mathcal{F}} C(\mathcal{J}, F_i)$, where $C(\mathcal{J}, F_i)$ follows Equation (3). Then the quality of $\mathcal{D}$ w.r.t. $\mathcal{F}$ is measured as*

$$Q(\mathcal{D}) = \frac{|C(\mathcal{J}, \mathcal{F})|}{|\mathcal{J}|}, \tag{4}$$

Intuitively, the quality is measured as the portion of the records that are correct among all the FDs (i.e., $C(\mathcal{J}, \mathcal{F}) = \bigcap_{\forall F_i \in \mathcal{F}} C(\mathcal{J}, F_i)$) on the instance.

Impact of join on data quality. To provide high-quality data for purchase, a naive solution is to clean the inconsistent data off-line before processing any data acquisition request online. However, this solution in incorrect, since *join can change data quality*: high-quality datasets may become low-quality after join, and vice versa. Example 2.2 shows an example of such quality change.

Table 1: An example: join of two high-quality data instances becomes low-quality

TID	A	B	C
t_1	a_1	b_2	c_4
t_2	a_1	b_2	c_1
t_3	a_1	b_2	c_2
t_4	a_1	b_3	c_3

(a) $D_1 : A \rightarrow B$
$(Q(D_1) = 0.75)$

TID	C	D	E
t_1	c_1	d_1	e_1
t_2	c_1	d_1	e_1
t_3	c_2	d_1	e_2
t_4	c_3	d_1	e_2
t_5	c_3	d_1	e_2

(b) $D_2 : D \rightarrow E$
$Q(D_2) = 0.6$

TID	A	B	C	D	E
t_1	a_1	b_2	c_1	d_1	e_1
t_2	a_1	b_2	c_1	d_1	e_1
t_3	a_1	b_2	c_2	d_1	e_2
t_4	a_1	b_3	c_3	d_1	e_2
t_5	a_1	b_3	c_3	d_1	e_2

(c) $D_1 \bowtie D_2$: $A \rightarrow B$, $D \rightarrow E$
$Q(D_1 \bowtie D_2) = 0.2$

Example 2.2. *In Table 1 (a) and (b), we display two datasets with good quality. The correct records in D_1 include $\{t_1, t_2, t_3\}$. Thus $Q(D_1) = 0.75$. Similarly, the correct records in D_2 include $\{t_3, t_4, t_5\}$. Thus $Q(D_2) = 0.6$. The join of D_1 and D_2 (Table 1 (c)) consists of 5 tuples. Apparently, $C(D_1 \bowtie D_2, A \rightarrow B) = \{t_1, t_2, t_3\}$, while $C(D_1 \bowtie D_2, D \rightarrow E) = \{t_3, t_4, t_5\}$. Only record t_3 is correct for both FDs $A \rightarrow B$ and $D \rightarrow E$. Therefore, $Q(D_1 \bowtie D_2) = 0.2$. Apparently, the quality of join results becomes much lower than that of D_1 and D_2.*

Example 2.2 shows that performing data cleaning before join is not acceptable. Thus data quality has to be measured on the join result online during data acquisition.

2.5 Problem Definition

Intuitively, the data shopper prefers to purchase the high-quality, affordable data instances that can deliver good join informativeness, and most importantly, the best correlation. We assume the data samples have been obtained from the data marketplace. Formally, given a set of data samples $\mathcal{D} = \{D_1, \dots, D_n\}$ collected from the data marketplace, a set of source instances $\mathbb{S}$ with source attributes $\mathcal{A}_\mathcal{S}$, a set of target attributes $\mathcal{A}_\mathcal{T}$, and the budget B for data purchase, the data acquisition problem is defined as the following: find a set of database instances $\mathbb{T}$ such that

$$\begin{aligned} \underset{\mathbb{T}}{\text{maximize}} \quad & CORR(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T}) \\ \text{subject to} \quad & \forall T_i \in \mathbb{T}, \exists D_j \in \mathcal{D} \ s.t. \ T_i \subseteq D_j, \\ & \sum_{T_i \in \mathbb{S} \cup \mathbb{T}} JI(T_i, T_{i+1}) \leq \alpha, \\ & Q(\mathbb{T}) \geq \beta, \\ & p(\mathbb{T}) \leq B, \end{aligned}$$

where α and β are the user-specified thresholds for join informativeness and quality respectively. We will discuss how to guide the data shopper to specify appropriate parameter values for α and β in Section 7.1. The output $\mathbb{T}$ will guide DANCE to suggest the data shopper of the SQL queries for data purchase from the data marketplaces.

3. DATA SAMPLING AND ESTIMATION

In this section, we discuss how to estimate join informativeness, data quality, and correlation based on samples. We use the correlated sampling [35] method to generate the samples from the data marketplace. General speaking, for any tuple $t_i \in D_1$, let $t_i[J]$ be its join attribute value. The record t_i is included in the sample S_1 if $hash(t_i[J]) \leq p_1$, where $hash()$ is the hash function that maps the join attribute values uniformly into the range $[0, 1]$, and p_1 is the sampling rate. The sample S_2 of D_2 is generated in the same fashion.

For a pair of data instances D_1 and D_2, let S_1 and S_2 be the samples of D_1 and D_2 by using correlated sampling. Then the join informativeness $JI(D_1, D_2)$ is estimated as:

$$\hat{JI}(D_1, D_2) = JI(S_1, S_2),$$

where $JI()$ follows Equation (1).

Next, we have Theorem 3.1 to show that the estimated join informativeness is unbiased and is expected to be accurate.

Theorem 3.1. *Let D_1 and D_2 be two data instances, and S_1 and S_2 be the samples of D_1 and D_2 by using correlated sampling with the same sampling rate. Then the expected value of the estimated join informativeness $E(JI(S_1, S_2))$ must satisfy that $E(JI(S_1, S_2)) = JI(D_1, D_2)$.*

The correctness of Theorem 3.1 relies on the fact that each tuple is sampled with the same probability. We present the proof of Theorem 3.1 in our full paper [23].

One weakness of the correlated sampling for estimation of the correlation and data quality is that the size of join result from samples can be extremely large, especially for the join of a large number of data instances. Note that this is not a problem for estimation of join informativeness as it only deals with 2-table joins. To deal with the large join result, we design the *correlated re-sampling* method by adding a second-round sampling of the join results. Intuitively, given a set of data instances $(D_1, \dots, D_p)$, for any intermediate join result IJ, if its size exceeds a user-specified threshold η, we sample IJ' from IJ by using a fixed re-sampling rate, and use IJ' for the following joins. In this way, the size of the intermediate join result is bounded.

For the sake of simplicity, for now we only focus on the re-sampling of 3-table joins (i.e., $D_1 \bowtie D_2 \bowtie D_3$). Let S_1, S_2 and S_3 be the samples of D_1, D_2, and D_3 respectively. Let S'_{12} denote the re-sampling result of $S_1 \bowtie S_2$ if its size exceeds η, or $S_1 \bowtie S_2$ otherwise. The estimated correlation and quality are

$$\widehat{CORR}_{D_1 \bowtie D_2 \bowtie D_3}(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T}) = CORR_{S'_{1,2} \bowtie S_3}(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T}),$$

and

$$\hat{Q}(D_1, D_2, D_3) = Q(S'_{1,2}, S_3).$$

The estimation can be easily extended to the join paths of arbitrary length, by applying sampling on the intermediate join results. Next, we show that the correlation and data quality estimation by using correlated re-sampling is also unbiased.

Theorem 3.2. *Given a join path (D_1, D_2, D_3), let $S'_{1,2}$ be the sample of $S_1 \bowtie S_2$. It must be true that the expected value of the estimated correlation $E(CORR_{S'_{1,2} \bowtie S_3})$ satisfies the following:*

$$E(CORR_{S'_{1,2} \bowtie S_3}(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T})) = CORR_{D_1 \bowtie D_2 \bowtie D_3}(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T}),$$

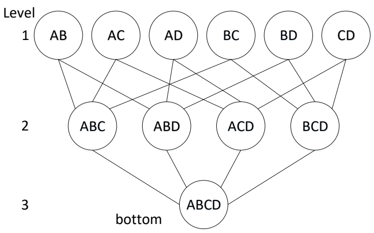


Figure 3: An example of attribute set lattice

and the expected value of the estimated quality $E(Q(S'_{1,2}, S_3))$ *satisfies the following:*

$$E(Q(S'_{1,2}, S_3)) = Q(D_1, D_2, D_3),$$

where $\mathcal{A}_\mathcal{S}$ *and* $\mathcal{A}_\mathcal{T}$ *are the source and target attribute sets.*

The correctness of Theorem 3.2 is similar to that of Theorem 3.1. We include the proof of Theorem 3.2 in our full paper [23] due to the space limit. We must note that the estimation is unbiased, regardless of the value of η.

4. OFF-LINE PHASE: CONSTRUCTION OF JOIN GRAPH

In this section, we present the concept of *join graph*, the main data structure that is used for our search algorithm (Section 5). First, we define the *attribute set lattice* of a single data instance.

Definition 4.1. **[Attribute Set Lattice (AS-lattice)]** *Given a data instance* D *that has* m *attributes* $\mathcal{A}$*, it corresponds to an* attribute set lattice *(AS-lattice)* $\mathcal{L}$*, in which each vertex corresponds to a unique attribute set* $\mathcal{A}' \subseteq \mathcal{A}$*. For the vertex* $v \in \mathcal{L}$ *whose corresponding attribute set is* $\mathcal{A}'$*, it corresponds to the projection instance* $\pi_{\mathcal{A}'}(D)$*. Given two lattice vertices* v_1, v_2 *of* $\mathcal{L}$*,* v_2 *is* v_1*'s* child *(and linked by an edge) if (1)* $\mathcal{A}_1 \subseteq \mathcal{A}_2$*, and (2)* $|\mathcal{A}_2| = |\mathcal{A}_1| + 1$*, where* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are the corresponding attribute sets of* v_1 *and* v_2*.* v_1 *is an* ancestor *of* v_2 *(linked by a path in* $\mathcal{L}$*) if* $\mathcal{A}_1 \subset \mathcal{A}_2$*.* v_1 *and* v_2 *are* siblings *if they are at the same level of* $\mathcal{L}$*.*

Given D and its m attributes $\mathcal{A}$, the height of the attribute set lattice of D is $m-1$. The bottom of the lattice contains a single vertex, which corresponds to $\mathcal{A}$, while the top of the lattice contains $\binom{m}{2}$ vertices, each corresponding to a unique 2-attribute set. Figure 3 shows an example of the attribute set lattice of an instance of four attributes $\{A, B, C, D\}$. In general, given an instance of m attributes, its attribute set lattice consists of $\binom{m}{2} + \ldots + \binom{m}{m} = 2^m - m - 1$ vertices.

Next, we define *join graph*. In the following discussion, we use $\mathsf{AS}(v)$ to denote the attribute set that the vertex v corresponds to.

Definition 4.2. **[Join Graph]** *Given a set of data instances (samples)* $\mathcal{D} = \{D_1, \ldots, D_n\}$*, it corresponds to an undirected, weighted, two-layer* join graph G*:*

(1) **Instance layer (I-layer)***: each vertex at this layer, called* instance vertex *(I-vertex), represents a data instance* $D_i \in \mathcal{D}$*. There is an edge* e_{ij}*, called* I-edge*, between any two I-vertices* v_i *and* v_j *if* $\mathsf{AS}(v_i) \cap \mathsf{AS}(v_j) \neq \emptyset$*.*

(2) **Attribute set layer (AS-layer)***: for each I-vertex* v_i*, it projects to a set of vertices, called* attribute set vertex *(AS-vertex), at the AS-layer, each corresponding to an unique attribute set of the instance* D_i*. The AS-vertices that was projected by the same I-vertex construct the AS-lattice (Def. 4.1). Each AS-vertex* v_i *is associated with a price* p_i*. For any two AS-vertices* v_i *and* v_j *that were projected by different I-vertices and* $\mathsf{AS}(v_i) \cap \mathsf{AS}(v_j) \neq \emptyset$*, there is an edge* e_{ij}*, called* AS-edge*, that connects* v_i *and* v_j*. Each AS-edge is associated with a pair* $(J, w_{i,j})$*, where* $J = \mathsf{AS}(v_i) \cap \mathsf{AS}(v_j)$*, and the weight* $w_{i,j}$ *is the join informativeness of* D_i *and* D_j *on the join attributes* J*.*

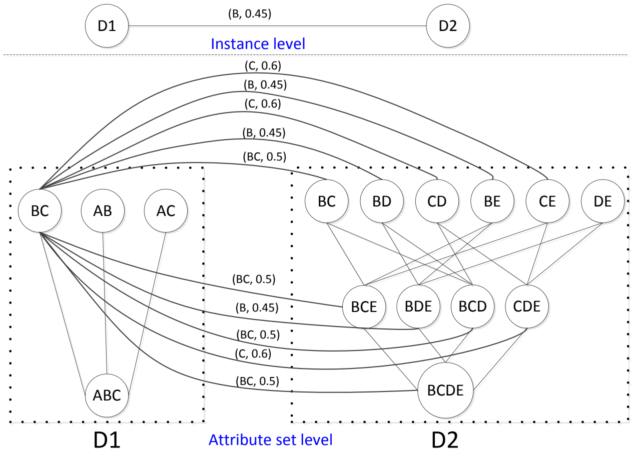


Figure 4: An example of a join graph for two instances $D_1(ABC)$ and $D_2(BCDE)$. Dotted rectangles represent tables. Only the node BC of instance D_1 has all edges. The edges of other nodes are omitted for simplicity.

Definition 4.2 only specifies the weight of AS-edges. Next, we define the weight of I-edges. For any two I-vertices v_i and v_j, let AE be the set of AS-edges $\{(v_k, v_l)\}$ where v_k (v_l, resp.) is an AS-vertex that v_i (v_j, resp.) projects to. The weight of the I-edge (v_i, v_j) is defined as the $w_{i,j} = \min_{(v_k,v_l) \in AE} w_{k,l}$.

The I-layer can be constructed from the schema information of datasets in the marketplaces. Many existing marketplace platforms (e.g., [3, 2]) provide such schema information. AS-layer will be constructed from the data samples obtained from the marketplace. Intuitively, there exists the trade-off between accuracy and cost. We will discuss this trade-off in Section 7.3.

Given a set of n data instances $\mathcal{D}$, its join graph contains n instance vertices and $\Sigma_{i=1}^{n}(2^{m_i} - m_i - 1)$ AS-vertices, where m_i denotes the number of attributes in D_i. Figure 4 shows an example of the join graph. An important property of the join graph is that all the AS-edges that connect the AS-vertices in the same instances with the same join attributes always have the same weight (i.e., the same join informativeness). For instance, the edges ($D_1.AB$, $D_2.BC$) and ($D_1.BC$, $D_2.BD$) have the same weight. Formally,

Property 4.1. *For any two AS-edges* (v_i, v_j) *and* (v'_i, v'_j)*, if* v_i *and* v'_i *were associated with the same I-vertex, as well as for* v_j *and* v'_j*, and* $\mathsf{AS}(v_i) \cap \mathsf{AS}(v_j) = \mathsf{AS}(v'_i) \cap \mathsf{AS}(v'_j)$ *(i.e., they have the same join attributes), then the two edges* (v_i, v_j) *and* (v'_i, v'_j) *are associated with the same weight.*

Property 4.1 is straightforward by following the definition of the join informativeness. For example, consider the join graph in Figure 4, the edge $(D_1.BC, D_2, BD)$ has the same weight as $(D_1.BC, D_2, BDE)$, as well as $(D_1.AB, D_2, BE)$ (not shown in Figure 4). Property 4.1 is in particular useful since it reduces the complexity of graph construction to exponential to the number of join attributes, instead of exponential to the number of all attributes. We will also make use of this property during graph search (Section 5) to speed up the search algorithm. We must note that the join informativeness does not have the *monotone* property (i.e., the join on a set of attributes J has higher/lower join informativeness than any subset $J' \subset J$). As shown in Figure 4, the join informativeness of join attributes BC is higher than that of join attribute B, but lower than that of the join attribute C.

Given a join graph, the source and target attributes $\mathcal{A}_\mathcal{S}$ and $\mathcal{A}_\mathcal{T}$ can be represented as special vertices in the join graph, formally defined below.

DEFINITION 4.3. **[Source/target vertex sets]** *Given the source instances $\mathbb{S}$, the source attributes $\mathcal{A}_\mathcal{S}$, and the target attributes $\mathcal{A}_\mathcal{T}$, and a join graph G. An instance vertex $v \in G$ is a* source *I-vertex if its corresponding instance $D \in \mathbb{S}$. An instance vertex $v \in G$ is a* target *I-vertex if its corresponding instance D is a source instance in $\mathbb{S}$ that contains at least one target attribute $A \in \mathcal{A}_\mathcal{T}$. A set of AS-vertices VS is a* source *(*target*, resp.) AS-vertex set if $\cup_{\forall v_i \in VS} \mathsf{AS}(v_i) = \mathcal{A}_\mathcal{S}$ ($\mathcal{A}_\mathcal{T}$, resp.). In other words, the AS-vertex set covers all source (target, resp.) attributes.*

Since some attributes may appear in multiple instances, there exist multiple source/target AS-vertex sets. Based on the join graph and the source/target AS-vertex sets, the data acquisition problem is equivalent to finding a subgraph named *target graph* in the join graph, which corresponds to the instances for purchase. Next, we formally define the *target graph*.

DEFINITION 4.4. **[Target Graph]** *Given a join graph G, a set of source AS-vertex sets $\mathcal{SV}$, and a set of target AS-vertex sets $\mathcal{TV}$, a connected sub-graph $TG \subseteq G$ is a* target graph *if there exists a source AS-vertex set $SV \in \mathcal{SV}$ and a target AS-vertex set $TV \in \mathcal{TV}$ such that TG contains both SV and TV.*

The target graph requires that it covers all source and target attributes. Next, we define the price, weight, and the quality of the target graph. Given a target graph TG, the price of TG is defined as $p(TG) = \sum_{\forall_{v_i \in TG}} p_i$. The *weight* of TG is defined as: $w(TG) = \sum_{\forall (v_i, v_j) \in TG} w(i, j)$. And the *quality* of TG, denoted as $Q(TG)$, is defined per Equation 4. Based on these definitions, now the data purchase problem can be mapped to the following graph search problem.

Problem statement Given a join graph G, a set of source AS-vertex sets $\mathcal{SV}$, a set of target AS-vertex sets $\mathcal{TV}$, and a budget B, find the *optimal target graph* (OTG) $G^* \subseteq G$ that satisfies the following constraint:

$$\begin{aligned} \text{Maximize} \quad & CORR(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T}) \\ \text{Subject to} \quad & w(G^*) \leq \alpha, \\ & Q(G^*) \geq \beta, \\ & p(G^*) \leq B, \end{aligned} \tag{5}$$

where α and β are user-specified threshold for join informativeness and quality. We have the following theorem to show the complexity of the graph search problem.

THEOREM 4.1. *The OTG search problem is NP-hard.*

The correctness of Theorem 4.1 follows the fact that $CORR(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T})$ is a submodular function. According to [36, 21], it is NP-hard to maximize a submodular function under a matroid constraint. The details of the proof are included in the full paper [23].

5. ONLINE PHASE: DATA ACQUISITION

Given the acquisition request $(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T})$ with the source data $\mathbb{S}$, the brute-force approach is to search all possible target graphs in the join graph to find the best one (i.e., $\mathcal{A}_\mathcal{S}$ and $\mathcal{A}_\mathcal{T}$ have the largest correlation). Obviously this approach is not scalable, given the fact that there is a large number of data instances in the data marketplace, and each data instance has exponentially many choices of attribute sets. Therefore, we design a heuristic algorithm based on Markov chain Monte Carlo (MCMC). The intuition is that with fixed $(\mathcal{A}_\mathcal{S}, \mathcal{A}_\mathcal{T})$, a large target AS-vertex set usually renders the join with small correlation and high join informativeness over longer join paths. Our algorithm consists of two steps. First, we find the minimal weighted I-layer graphs (I-graphs) (i.e., minimal join informativeness and thus higher correlation). Second, we find the local optimal target graph at the AS-layer (AS-graphs) from the minimal weighted I-graphs. Next, we present the details of the two steps.

5.1 Step 1: Find Minimal Weighted Graphs (I-graphs) at I-layer

Given the source and target vertices, our graph search problem can be transformed to the classic Steiner tree problem that searches for a minimal tree that connects the source and target vertices. The Steiner tree problem is NP-hard. The complexity of the approximation algorithm [34] is quadratic to the number of nodes, which may not be acceptable for large graphs. In this paper, we design a heuristic algorithm whose complexity is logarithmic to the number of nodes in the graph. Our algorithm extends the approximate shortest path search algorithm [13]. The key idea is to randomly pick a set of I-vertices as the *landmarks*. For each I-vertex in G, our algorithm pre-computes and stores the minimal weighted paths to these landmarks, by using the shortest path search algorithm [13]. Then given the source and target AS-vertex sets $\mathcal{A}_\mathcal{S}$ and $\mathcal{A}_\mathcal{T}$, for each landmark v_m, the algorithm constructs a graph by connecting each vertex in $\mathcal{A}_\mathcal{S} \cup \mathcal{A}_\mathcal{T}$ and v_m, via their minimal weighted paths. The output I-graph is constructed as the union of all the minimal weighted paths. If the total weight of the I-graph exceeds α, there does not exist a target graph that satisfies the join informativeness constraint. The algorithm returns no output for this case.

5.2 Step 2: Find Optimal Target Graphs (AS-graphs) at AS-layer

Based on the data instances selected by Step 1, Step 2 further selects the projection attributes of these instances by searching at the AS-layer of the minimal weighted I-graph. Algorithm 1 shows the pseudo code of Step 2. The key idea of Step 2 is to generate a sample of the target graph at AS-layer iteratively by replacing the join attribute set of one edge $e_{i,j}$ with a different join attribute set. The algorithm runs ℓ iterations and keeps the target graph of the largest correlation between the source and target vertices (Line 11 - 14). Some target graphs may not satisfy the constraints on weights, quality, and/or price. For each new target graph, we first check if it satisfies these constraints (Line 8). After that, we use a Markov Chain Monte Carlo (MCMC) process to generate target graphs with high correlation, so as to maximize the utility of the output. In particular, for each graph, the algorithm randomly picks an edge $e_{i,j}$ (Line

Algorithm 1: FindJoinTree_AttSet(): find optimal target graph at attribute set layer

```
Require: A minimal weighted I-graph $\mathcal{IG}$
Ensure: A target graph $G^*$ at the AS-layer
 1: $G^* = NULL$
 2: $Max = 0$
 3: $TG = \mathcal{IG}$
 4: for $i = 1$ to $\ell$ do
 5:   Randomly pick an edge $e_{i,j} \in TG$
 6:   Randomly pick a different edge $e'_{i,j}$ between $(v_i, v_j)$
 7:   Let $TG'$ be the new target graph
 8:   if $p(TG') \leq B \wedge w(TG') \leq \alpha \wedge Q(TG') \geq \beta$ then
 9:     if accept $e'_{i,j}$ by probability $\min(1, \frac{CORR(TG')}{CORR(TG)})$ then
10:       $TG = TG'$
11:       if $CORR(TG) > Max$ then
12:         $G^* = TG$
13:         $Max := CORR(G^*)$
14:       end if
15:     end if
16:   end if
17: end for
18: return $G^*$
```

5), which represents the join between two instances D_i and D_j. From all the possible join attributes between D_i and D_j, the algorithm randomly picks one, which corresponds to the AS-edge $e'_{i,j}$, and replaces $e_{i,j}$ with $e'_{i,j}$ by the acceptance probability $\min(1, \frac{CORR(TG')}{CORR(TG)})$ (Line 9). Intuitively, the target graph of high correlation is accepted with high probability.

5.3 Complexity Analysis

The complexity of Step 1 is $O(k(|\mathcal{A}_\mathcal{S}| + |\mathcal{A}_\mathcal{T}|) \log_2 n)$, where k is the average size of the minimal weighted graphs at the I-layer, and n is the number of data instances on the marketplace (i.e., the number of vertices at the I-layer of the join graph). In the experiments, we observe that k is much smaller than n, especially when n is large. The complexity of Step 2 is $O(\ell C_J)$, where ℓ is the number of iterations, and C_J is the average join cost of the instances in the I-graph returned by Step 1. Note that C_J depends on both the size of I-graphs, and the size of each instance in the I-graph. The total complexity of the graph search algorithm is dominated by Step 2, which is $O(\ell C_J)$.

6. EXPERIMENTS

6.1 Experimental Setup

Implementation & testbed. We implement the algorithms in *Python*. All the experiments are executed on a machine with 2 × Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz, 12 cores, 24 processors, and 128 GB memory.

Datasets. We use three datasets, including TPC-E[1], TPC-H[2] benchmark datasets, and IMDB dataset[3], in our experiments. The details of the three datasets are shown in Table 2. The longest path of TPC-H and TPC-E join graphs are of length 8 and 9 respectively, and 2 for IMDB join graph. IMDB join graph include seven nodes. Six nodes form a clique. The remaining single node (corresponding to the table *name*) connects with only one node (corresponding to the table *principals*) in the clique.

Functional Dependency (FD). We implemented the FD discovery algorithm in [15] to find the FDs. The number of discovered FDs is shown in Table 2.

Data quality. To introduce inconsistency into the tables, we modified 30% of records of 6 tables in TPC-H dataset (except *region* and *Nation* tables), 20 out of 29 tables in TPC-E dataset, and all tables in IMDB dataset.

Data acquisition queries. For TPC-H and TPC-E datasets, we define three queries Q_1, Q_2 and Q_3 for short, medium, and long join paths respectively. Q_1, Q_2 and Q_3 for TPC-H dataset are of join path length 2, 3, and 5, while for TPC-E dataset, Q_1, Q_2 and Q_3 are of join path length 3, 5, and 8. Furthermore, to measure the impact of instance size on the performance of data acquisition, the source/target attributes of Q_1, Q_2, Q_3 for TPC-H dataset are associated with instances of various sizes: the source and target attributes of Q_1 are associated with the instances of medium- (150K records) and large-size (1.5M records) respectively; the source and target attributes of Q_2 are associated with small- (25 records) and medium-size (800K records) tables; and the source and target attributes of Q_3 are associated with the tables of large (1.5M records) and small (5 records) sizes. The two queries Q_1 and Q_2 for IMDB dataset are defined to measure the impact of number of join-able tables of the source/target attributes on the performance of data acquisition. For Q_1, both source and target attributes can join with 5 tables. For Q_2, the source attribute belongs to the table that only has one join-able partner, while the target attribute can join with 5 tables. The source, target, and semantics of all the queries are shown in Table 3.

Pricing function & budget. We use the well-known entropy-based pricing function [19]. To simulate various budget settings, for each acquisition query, first, we measure the *lowerbound LB* and *upperbound UB* of the total budget as the minimum and maximum price of all possible paths between source and target nodes. We define the shopper's budget as $r \times UB$, where $r \in (0, 1]$ is the *budget ratio*. Intuitively, the larger r is, the more budget that the shopper is equipped for data purchase. We require that $r \times UB \geq LB$, i.e., the shopper can afford to purchase the instances of at least one join path in the join graph.

Evaluation metrics. To evaluate the performance of our sampling-based heuristic algorithm, we compare the correlation of the identified data instances by our algorithm and two optimal algorithms, namely the *local optimal (LP)* algorithm and the *global optimal (GP)* algorithm. Both LP and GP algorithms are the brute-force algorithms that enumerate all paths to find the one of the highest correlation. LP algorithm uses the samples as the input, and the GP algorithm uses the original datasets as the input. For all the algorithms, we measure the real correlation, not the estimated value. Let X_{OPT} and X be the correlation of the output by the (LP/GP) optimal approach and our heuristic approach respectively. The *correlation difference* of is measured as $CD = \frac{X_{OPT} - X}{X_{OPT}}$. Intuitively, the smaller correlation difference is, the more accurate our heuristic algorithm is.

Baseline. As this is the first work on data acquisition for data marketplaces, we do not find any state-of-the-art work

[1] http://www.tpc.org/tpce/

[2] http://www.tpc.org/tpch/

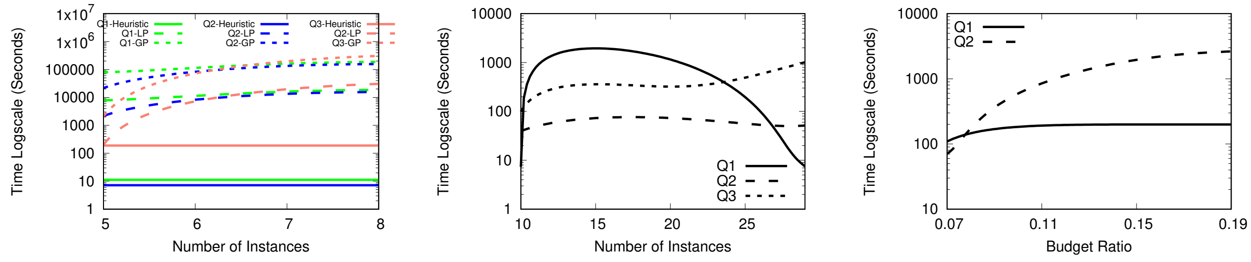
[3] https://www.imdb.com/interfaces/

Table 2: Dataset description (the tables that are of the minimum/maximum number of records/attributes are shown in a pair of parentheses).

	# of	Min. instance size	Max. instance size	Avg. instance size	Max. # of	Avg # of FDs	Total # of
	instances	(# of records)	(# of records)	(# of records)	attributes	per table	join edges
TPC-H	8	5 (Region)	6,000,000 (Lineitem)	1,082,655	20 (Lineitem)	39	11
TPC-E	29	4 (Exchange)	10,001,048 (Watchitem)	672,353	28 (Customer)	33	72
IMDB	7	866,657(Ratings)	30,007,771(Principals)	8,218,165	8(akas,basics)	27	16

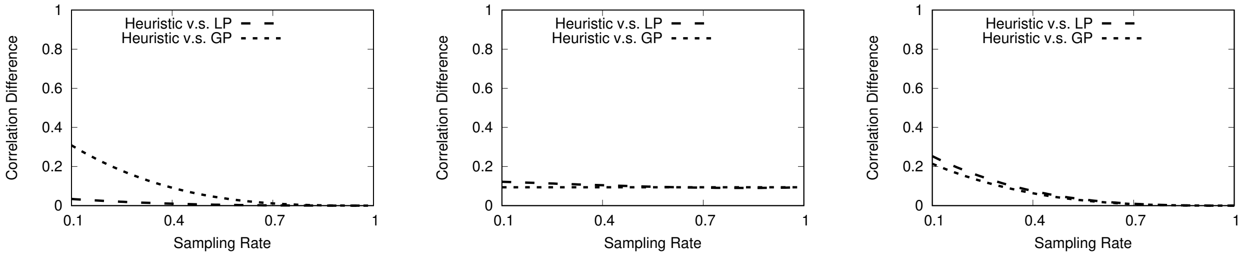
Table 3: Query description

Dataset	Query	Source	Target	Explanation
TPC-H	Q1	customer.account_balance	orders.clerk	link customers' account with responsible clerks
	Q2	nation.name	partsupp.availqty	link parts with the nation of their suppliers
	Q3	orders.total_price	region.name	associate orders' price with the origin region
TPC-E	Q1	address.ad_line1	customeraccount.ca_name	pair customers with addresses
	Q2	newseref.ni_id	watchlist.w_id	find news relevant to securities in customers' watchlist
	Q3	customertaxrate.tx_id	zipcode.zc_town	associate tax with zipcode
IMDB	Q1	crew.directors	ratings.averageRating	associate movie's ratings with director
	Q2	name.knownForTitles	akas.language	associate actors' famous movies with languages



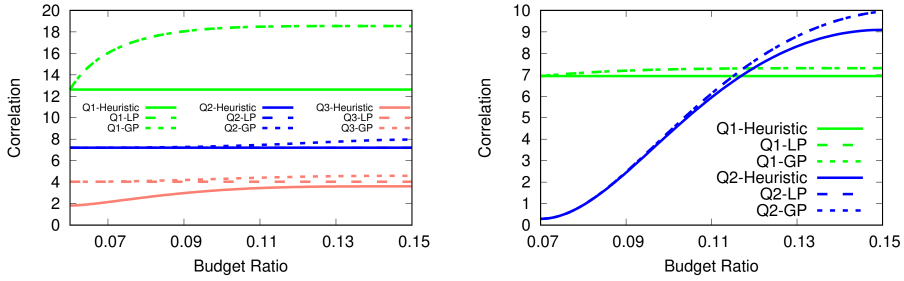
(a) TPC-H (b) TPC-E (c) IMDB

Figure 5: Time performance w.r.t. various # of instances and budget ratios



(a) Q_1 (b) Q_2 (c) Q_3

Figure 6: Correlation difference w.r.t. various sampling rate (TPC-H dataset)

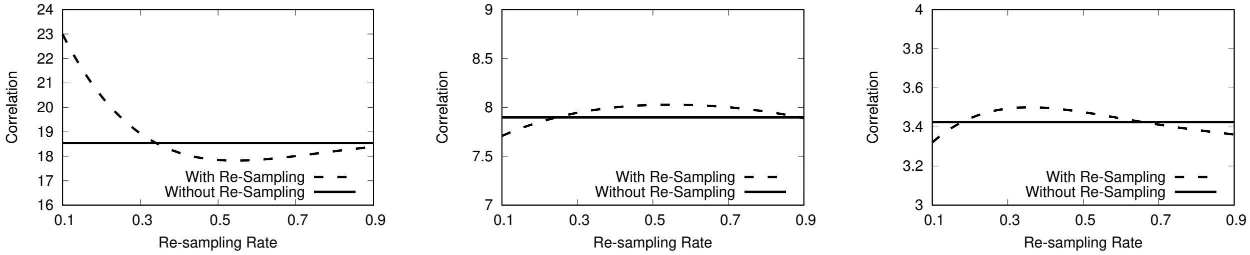


(a) TPC-H (b) IMDB

Figure 7: Correlation w.r.t. various budget ratio

Table 4: Comparison between our search algorithm and GREEDY algorithm (TPC-E dataset)

Query	Approach	Correlation	Quality	Join Informativeness	Price
Q_1	Our Approach	16.87	0.94	1.76	84.82
	GREEDY Algorithm	16.87	0.94	1.76	84.82
Q_2	Our Approach	15.51	0.96	2.03	104.52
	GREEDY Algorithm	5.85	0.67	3.78	175.03
Q_3	Our Approach	2.25	1	5.10	134.53
	GREEDY Algorithm	0	0	7.06	168.42



(a) Q_1 (b) Q_2 (c) Q_3

Figure 8: Correlation with and without re-sampling w.r.t. various re-sampling rates (TPC-H dataset)

Table 5: Comparison between data acquisition with DANCE and purchase from the data marketplace directly (TPC-H dataset, budget ratio=0.13)

Query	Approach	Correlation	Quality	Join Informativeness	Price
Q_1	With DANCE	12.51	0.07688	0.8906	60.65
	Purchase from data marketplace	18.18	0.3819	0.8268	63.12
Q_2	With DANCE	7.216	0.3968	1.248	59.78
	Purchase from data marketplace	7.965	0.4244	1.040	103.2
Q_3	With DANCE	3.609	0.08632	2.063	100.5
	Purchase from data marketplace	4.592	0.1026	2.104	106.4

Table 6: Comparison between data acquisition with DANCE and purchase from the data marketplace directly (IMDB dataset, budget ratio=0.14)

Query	Approach	Correlation	Quality	Join Informativeness	Price
Q_1	With DANCE	6.94	1	0.946	61.94
	Purchase from data marketplace	7.31	1	1.798	82.29
Q_2	With DANCE	9.094	1	1.715	111.391
	Purchase from data marketplace	9.957	1	2.663	136.824

to compare with. Thus we design the GREEDY algorithm that picks the edge in the join graph with the minimum weight at each step of graph traversal, until the target is reached. The GREEDY algorithm allows backward traversal if the current path is not reachable to the target.

6.2 Scalability

First, we measure the time performance of our heuristic algorithm with various number of data instances. In Figure 5 (a), we compare the time performance of our heuristic algorithm against LP and GP on TPC-H data. First, we observe that our heuristic algorithm is significantly more efficient than the two optimal algorithms, especially when n (the number of instances) is large. For example, when $n = 8$, our heuristic algorithm can be 2,000 times more efficient than LP, and 20,000 times more efficient than GP. We are aware that for Q_3, the heuristic algorithm has comparable time performance to LP when $n = 5$. This is because for this case, there is only one single I-graph that connects the source and target vertices, which leads to the same search space of our heuristic algorithm and LP. Second, we observe that the time of both LP and GP increase with the growth of n. This is not surprising since the number of I-graphs that connect the source and target vertices increase with the growth of n. However, the time performance of the heuristic algorithm keeps stable when n increases. This is because the minimal weighted I-graph identified by Step 1 (Section 5.1) keeps.

Figure 5 (b) shows the time performance of our approch on TPC-E dataset. Since the two optimal algorithms do no halt within 10 hours on this large dataset, we only show the time performance of our heuristic algorithm. An important observation is that the time performance does not increase with the growth of number of instances for some queries (e.g., Q_1 and Q_3). This is because while the time performance of our heuristic algorithm depends on the I-graph size, the increase of the number of instances does not necessarily increase the I-graph size. For example, consider Q_1 in Figure 5 (b), the I-graph size drops 57% when the number of instances changes from 15 to 29.

We also measure the time performance of our heuristic algorithm with regard to various budget ratios. Figure 5

(c) shows the time performance of our heuristic algorithm with various budget ratios on IMDB dataset. We vary the budget ratio from 0.07 to 0.19, where 0.07 is the minimum budget ratio that can find at least one solution for either of the two queries Q_1 and Q_2, while 0.19 is the budget ratio that can cover all the join paths in the search space of these two queries. From the results, we observe that DANCE takes more time when the budget ratio grows. However, the time performance may keep stable when the budget ratio is sufficiently large. For example, when the budget ratio is larger than 0.11, the time performance of Q_1 keeps unchanged. This is because when the budget ratio is 0.11 or larger, the search space of Algorithm 1 remains the same. We also measure the time performance on TPC-H and TPC-E dataset with various budget ratios, and have similar observation. Due to the space limit, we present the detailed results in our full paper [23].

6.3 Correlation

First, we measure the difference of the correlation of source and target attribute sets in the datasets returned by our heuristic algorithm and the two optimal algorithms with regard to various sampling rates. Intuitively, we want the correlation difference to be close to 0. The results are shown in Figure 6. First, we notice that the correlation difference is very small. In all cases, it never exceeds 0.31. Recall that in Figure 5, our heuristic algorithm can be 20,000 times more efficient than the optimal methods. This demonstrates that our method can efficiently find the datasets of correlation that is comparable to the optimal results. Second, we observe that the correlation difference decreases when the sampling rate grows. This is straightforward as more samples lead to more accurate correlation estimation.

Second, we measure the correlation between source and target attributes in the data acquisition result returned by our heuristic algorithm, as well as in the two optimal algorithms, on TPC-H and IMDB datasets. The comparison results are displayed in Figure 7. First, we notice that the correlation by our heuristic algorithm is close to that of both optimal algorithms. In all cases on both datasets, the difference is at most 5.9 (the maximum correlation is around 19). Second, with the increase of the budget ratio, the correlation of the results by all the three algorithms gradually rise (i.e., the correlation gets stronger). This is straightforward, as higher budget can afford to purchase more data with better utility. We do not show the correlation measurement results on TPC-E dataset, due to the long execution time of the GP algorithm.

Third, we compare the performance of our heuristic algorithm with GREEDY algorithm in terms of correlation as well as quality, join informativeness and price of the acquisition results on TPC-E dataset. The results are shown in Table 4. For queries of short path (e.g., Q_1), the GREEDY algorithm finds the same acquisition results as our approach. For queries of longer path (e.g., Q_2), the correlation, quality, join informativeness and price of the acquisition results by the GREEDY algorithm is much worse than our approach. For the worst case, the GREEDY algorithm returns a path whose join result is totally meaningless (e.g, Q_3).

We also measure the impacts of re-sampling on the correlation by changing the re-sampling rate on TPC-H dataset. The result is presented in Figure 8. We observe that the correlation with re-sampling oscillates around the correlation without re-sampling. The difference gradually reaches 0 with the growth of the re-sampling rate. Overall, the estimated correlation with re-sampling is accurate. The difference with the estimated correlation without re-sampling never exceeds 4.5.

6.4 Data Acquisition with DANCE vs. without DANCE

We compare the data acquisition results (correlation, data quality, join informativeness, and price) by DANCE with direct purchase from the data marketplace on TPC-H and IMDB datasets. We use the GP algorithm to find the data acquisition results on the data marketplace. The comparison result for TPC-H dataset is displayed in Table 5. First, we observe a large overlap between the data acquisition results returned by DANCE and directly from data marketplace. For instance, for Q_3, the two results has 91% overlap. Due to the space limit, we include the details of the acquisition results in the full paper [23]. Second, we observe that the correlation of the data acquisition results returned by DANCE is comparable to that returned from the data marketplace. It can be as high as 90% of the optimal result. Third, the join informativeness of the data acquisition by both DANCE and from the data marketplace is also close, which demonstrates the superiority of our correlated re-sampling method. The price of the data acquisition results returned by DANCE is always lower than that from the data marketplace. For example, it is 42% lower than the price of Q_2 directly from the data marketplace. This shows that DANCE is able to find the data of high utility (correlation) at a lower price. We acknowledge that in some cases, the quality of the data acquisition results returned by DANCE (e.g., Q_1) is significantly lower than from the marketplace directly, due to the error introduced by the sampling-based estimation. However, in most cases, the accuracy is still satisfying (Q_2 and Q_3). The comparison result for IMDB dataset is displayed in Table 6. The main observations are similar to the TPC-H dataset. We omit the detailed discussion here.

7. EXTENSIONS

7.1 Parameter Setup

Providing appropriate parameter values (e.g., the threshold of data quality and join informativeness) for the data acquisition requests is challenging for ordinary users who lack expertise and background in database management. DANCE can recommend the threshold settings to the users. These recommended threshold settings can be generated via optimal experimental design (OED) techniques [12]. In particular, given a set of design samples, each associated with a set of parameter values and the performance measure, OED aims to find a design of parameter settings that maximizes the performance measure in expectation. DANCE can collect the design samples either in the off-line phase by performing multiple trials on the data samples, or in the online phase by collecting users' data acquisition requests and their results. The optimal parameter settings can be learned by applying active learning on the collected design samples [38].

7.2 System Plug-in with More Settings

New data quality metrics. In this paper, we mainly consider FD-based metrics [15, 9]. Other metrics that measure data dependencies (e.g., CFDs [11] and CINDs [7]) can be easily adapted to DANCE by simply replacing FDs with these metrics. Our sampling approaches (Section 3) can estimate the quality for these metrics. Changing data quality metrics only impacts the calculation of data quality during search. Our search algorithm keeps unchanged.

New data pricing functions. In the paper we employ the entropy-based data pricing function [19], due to its arbitrage-free property. Other pricing functions [10, 22] can be easily plugged into DANCE for price calculation. Changing pricing functions only impacts the calculation of prices. It will not change our search algorithm.

7.3 Acquisition Cost Model

Purchasing data samples from the data marketplace in the off-line phase is not free. Apparently, purchasing more samples can better serve the data shoppers' acquisition needs. But it occurs higher cost. There exists the trade-off between the performance of data acquisition and the cost of sample purchase. Nevertheless, first, the cost of sampling purchase can be amortized over multiple data acquisition queries in the online phase. Second, sampling purchase is not necessarily a one-time process. DANCE can keep purchasing more samples from the data marketplace to improve its quality of data acquisition service. An interesting observation from our empirical study is that the sampling rate as small as 0.3 indeed can provide satisfying estimation accuracy. This observation can be used as the guidance to decide how many samples should be purchased.

7.4 Dealing with Data Updates

In practice, the data marketplace is highly volatile as new data are produced rapidly. When there are data updates in the marketplace, the join graph should be updated accordingly. For example, new data instances in the marketplace introduces new nodes and edges inserted into the join graph. This requires DANCE to purchase new samples from the data marketplace, which incurs monetary costs. Using outdated data for analysis can avoid the data acquisition costs, but the analysis results may be inaccurate. This raises the decision making problem of when to acquire new data from the marketplace and when to use existing data for data analysis. Intuitively, a data item that is likely outdated and significantly affects query result, as well as a data item that is likely used often in queries in the near future, should be purchased for join graph update. We can quantify the *reward* of new data in terms of data correlation, which can help to decide whether it is worth to purchase new data samples from the marketplace. Machine learning methods (e.g., reinforcement learning [24]) can be adapted to solve the decision making problem.

8. RELATED WORK

The concept of *data marketplace* is firstly formally defined in [6]. Kanza et al. [16] envision a geo-social data marketplace that facilitates the generation and selling of high-quality spatio-temporal data from people. Koutris et al. [19] propose a query-based data pricing model for the market, considering the complicated queries that involve conjunctive sub-queries. Ren et al. [30] focus on the joint problem of data purchasing and data placement in a cloud data market. None of them studies the correlation-driven purchase on the data marketplace. Balazinska et al. [6] propose the data pricing model in which the buyers are charged based on the queries. QueryMarket [18] demonstrates the superiority of such a query-based data pricing model in real-world applications.

A relevant line of work is to explore the databases that contain complex database schemas via joins. Intuitively, given a database with schema graph, in which two tables are specified as the source and destination tables, the problem is to find a valid join path between the source and destination. Procopiuc et al. [27] compute a probability for each join path that connects a given source table to a destination table. To speed up the exploration over the complex schema graph, Yang et al. [37] summarize the content of a relational database as a *summary graph*. The most relevant tables and join paths can be computed from the summary graph. It defines the importance of each table in the database as its stable state value in a random walk over the schema graph, where the transition probability is defined based on the entropies of table attributes. Zhang et al. [39] take the reverse-engineering approach. In particular, given a database D with schema graph G and an output table Out, the problem is to compute a join query Q that generates Out from D. Unlike these works that mainly focus on the join informativeness of data instances, we take multiple factors, including join informativeness, data quality, and price, into consideration for the data acquisition on the data marketplace.

Regarding the related work on subgraph search, the work on centerpiece subgraphs [33, 32], and on entity-relationship subgraphs [17] partially share with our goal: for a given set of query nodes and a budget constraint, output a connecting subgraph that meets the budget and maximizes some goodness measure. These work mainly focus on mining the patterns of some specific subgraphs, while we focus on finding the optimal subgraph with regards to the given constraints on informativeness, quality, and price.

9. CONCLUSION

In this paper, we study the data acquisition problem for correlation analysis in the data marketplace. We consider quality, join informativeness, and price issues of data acquisition, and model the data acquisition problem as a graph search problem under various constraints. We prove that the graph search problem is NP-hard, and design a heuristic algorithm based on Markov chain Monte Carlo (MCMC). Our experiment results demonstrate the efficiency of our method.

For the future, there are quite a few interesting research directions to explore. For example, instead of the best acquisition scheme, DANCE may recommend a number of acquisition options of the top-k scores to the data buyer, where the scores can be defined as a combination of correlation, data quality, join informativeness, and price. This raises the issues of how to define a fair score function, as well as the design of efficient top-k search algorithm when the score function is not be monotone. Furthermore, an important issue is the privacy of the data marketplaces. In particular, when DANCE is not fully trusted, how can the data shopper exchange the information of the source instances with DANCE in a privacy-preserving way, and find correlated instances in the data marketplace? How to verify if the data in the marketplace is genuine (i.e., it is not fabricated)?

10. REFERENCES

[1] Bdex. http://www.bigdataexchange.com/.

[2] Google bigquery data warehouse. https://cloud.google.com/bigquery/.

[3] Microsoft azure data marketplace. https://azuremarketplace.microsoft.com.

[4] Idc's worldwide semiannual big data and analytics spending guide taxonomy. `http://www.informationweek.com/big-data/`, 2016.

[5] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 68–79, 1999.

[6] M. Balazinska, B. Howe, and D. Suciu. Data markets in the cloud: An opportunity for the database community. *PVLDB*, 4(12):1482–1485, 2011.

[7] L. Bravo, W. Fan, and S. Ma. Extending dependencies with conditions. *PVLDB*, 4(11):243–254, 2007.

[8] F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.

[9] F. Chiang and R. J. Miller. A unified model for data and constraint repair. In *IEEE International Conference on Data Engineering*, pages 446–457, 2011.

[10] S. Deep and P. Koutris. The design of arbitrage-free data pricing schemes. In *International Conference on Database Theory*, pages 1–18, 2017.

[11] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Transactions on Database Systems (TODS)*, 33(2):6, 2008.

[12] V. Fedorov. Optimal experimental design. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):581–589, 2010.

[13] A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of ACM International Conference on Information and Knowledge Management*, pages 499–508, 2010.

[14] P. N. Hague, N. Hague, and C.-A. Morgan. *Market research in practice: How to get greater insight from your market.* Kogan Page Publishers, 2013.

[15] Y. Huhtala et al. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.

[16] Y. Kanza and H. Samet. An online marketplace for geosocial data. In *Proceedings of the ACM International Conference on Advances in Geographic Information Systems*, pages 10–13, 2015.

[17] G. Kasneci, S. Elbassuoni, and G. Weikum. Ming: mining informative entity relationship subgraphs. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 1653–1656, 2009.

[18] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Querymarket demonstration: Pricing for online data markets. *PVLDB*, 5(12):1962–1965, 2012.

[19] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Query-based data pricing. *Journal of the ACM*, 62(5):43, 2015.

[20] A. Kraskov and P. Grassberger. Mic: mutual information based hierarchical clustering. In *Information Theory and Statistical Learning*, pages 101–123. Springer, 2009.

[21] A. Krause and D. Golovin. Submodular function maximization., 2014.

[22] C. Li and G. Miklau. Pricing aggregate queries in a data marketplace. In *WebDB*, pages 19–24, 2012.

[23] Y. Li, H. Sun, B. Dong, and W. H. Wang. Cost-efficient data acquisition on online data marketplaces for correlation analysis (full version). Technical report, 2018. Available at `https://msuweb.montclair.edu/~dongb/publications/dacron-full.pdf`.

[24] Z. Li and T. Ge. Stochastic data acquisition for answering queries as time goes by. *PVLDB*, 10(3):277–288, 2016.

[25] S. K. Mitra and Y. Kuo. *Digital signal processing: a computer-based approach*, volume 2. McGraw-Hill New York, 2006.

[26] H. V. Nguyen, E. Müller, P. Andritsos, and K. Böhm. Detecting correlated columns in relational databases with mixed data types. In *Proceedings of the International Conference on Scientific and Statistical Database Management*, pages 30–42, 2014.

[27] C. M. Procopiuc and D. Srivastava. Database exploration using join paths. In *IEEE International Conference on Data Engineering*, pages 31–33, 2008.

[28] L. Qian, M. J. Cafarella, and H. Jagadish. Sample-driven schema mapping. In *Proceedings of the ACM International Conference on Management of Data*, pages 73–84. ACM, 2012.

[29] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[30] X. Ren et al. Joint data purchasing and data placement in a geo-distributed data market. *arXiv preprint arXiv:1604.02533*, 2016.

[31] S. Song and L. Chen. Efficient set-correlation operator inside databases. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 139–148. ACM, 2010.

[32] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 404–413, 2006.

[33] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 747–756, 2007.

[34] V. V. Vazirani. *Approximation algorithms.* Springer Science & Business Media, 2013.

[35] D. Vengerov, A. C. Menck, M. Zait, and S. P. Chakkappen. Join size estimation subject to filter conditions. *PVLDB*, 8(12):1530–1541, 2015.

[36] J. Vondrák. Submodularity in combinatorial optimization. 2007.

[37] X. Yang, C. M. Procopiuc, and D. Srivastava. Summary graphs for relational database schemas. *PVLDB*, 4(11):899–910, 2011.

[38] K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *Proceedings of the 23rd International Conference on Machine*

learning, pages 1081–1088. ACM, 2006.

[39] M. Zhang, H. Elmeleegy, C. M. Procopiuc, and D. Srivastava. Reverse engineering complex join queries. In *Proceedings of the ACM International Conference on Management of Data*, pages 809–820, 2013.