Dealer: An End-to-End Model Marketplace with Differential Privacy

Jinfei Liu
Emory University and Georgia Institute of Technology
jinfei.liu@emory.edu

Jian Lou
Emory University
jian.lou@emory.edu

Junxu Liu
Emory University
junxu.liu@emory.edu

Li Xiong
Emory University
lxiong@emory.edu

Jian Pei
Simon Fraser University
jpei@cs.sfu.ca

Jimeng Sun
Georgia Institute of Technology and UIUC
jsun@cc.gatech.edu

ABSTRACT

Data-driven machine learning has become ubiquitous. A marketplace for machine learning models connects data owners and model buyers, and can dramatically facilitate data-driven machine learning applications. In this paper, we take a formal data marketplace perspective and propose the first en**D**-to-end mod**e**l m**a**rketp**l**ace with diff**e**rential p**r**ivacy (*Dealer*) towards answering the following questions: *How to formulate data owners' compensation functions and model buyers' price functions? How can the broker determine prices for a set of models to maximize the revenue with arbitrage-free guarantee, and train a set of models with maximum Shapley coverage given a manufacturing budget to remain competitive*? For the former, we propose compensation function for each data owner based on Shapley value and privacy sensitivity, and price function for each model buyer based on Shapley coverage sensitivity and noise sensitivity. Both privacy sensitivity and noise sensitivity are measured by the level of differential privacy. For the latter, we formulate two optimization problems for model pricing and model training, and propose efficient dynamic programming algorithms. Experiment results on the real chess dataset and synthetic datasets justify the design of *Dealer* and verify the efficiency and effectiveness of the proposed algorithms.

PVLDB Reference Format:
Jinfei Liu. A Sample Proceedings of the VLDB Endowment Paper in LaTeX Format. *PVLDB*, 12(xxx): xxxx-yyyy, 2019.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

Machine learning has witnessed great success across various types of tasks and is being applied in an ever-growing number of industries and businesses. High usability machine learning models depend on a large amount of high-quality training data, which makes it evident that data is valuable. Recent studies and practices have approached the commoditization of data in various ways. A data marketplace sells data either in the direct or indirect (derived) forms. These data marketplaces can be generally categorized based on what they sell and their corresponding pricing mechanisms: 1) (raw) data-based pricing, 2) query-based pricing, and 3) model-based pricing.

Data marketplaces with data-based pricing sell datasets and allow buyers to access the data directly, e.g., Dawex [1], Twitter [3], Bloomberg [4], Iota [5], and SafeGraph [6]. Under these marketplaces, data owners have limited or no control over their data usages, which makes it challenging for the market to incentivize more data owners to contribute or results in a market lacking transparency. Also, it can be overpriced for buyers to purchase the whole dataset when they are only interested in particular information extracted from the dataset.

Data marketplaces with query-based pricing [24, 25], e.g., Google Bigquery [2], partially alleviate these shortcomings by charging buyers and compensating data owners on a per-query basis. The marketplace makes decisions about data usage restrictions (e.g., return queries with privacy protection [28]), compensation allocation, and query-based pricing. However, most queries considered by these marketplaces are too simplistic to support sophisticated data analytics and decision making.

Data marketplaces with model-based pricing [7, 13, 22] have been recently proposed. In [13], the authors focus on pricing a set of model instances depending on their model quality to maximize the revenue, while [22] considers how to allocate compensation in a fair way among data owners when their data are utilized for the model of *k*-nearest neighbors (*k*-NN). Notably, they are focused on either the data owner or the model buyer end of the marketplace but not both. Most recently, [7] approaches it in a relatively more complete perspective by studying two ends of the marketplace, and proposed strategies for the broker to set model usage charge from model buyers, and to distribute compensation to data owners. However, [7] oversimplifies the roles of the two end entities, i.e., data owners and model buyers. For example, data owners still have no means to control the way that their data is used, while model buyers do not have a choice over the quality of the model that best suits their demands and budgets.

Gaps and Challenges. Though efforts have been made to ensure the broker follows important market design principles in [7, 13, 22], how the model marketplace should respond to the needs of both data owners and model buyers are still understudied. It is therefore tempting to ask: how can we build an end-to-end marketplace dedicated to machine learning models, which can simultaneously

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

satisfy the needs of all three entities, i.e., data owners, broker, and model buyers. We summarize the gaps and challenges from the perspective of each entity as follows.

- *Data owners.* Under the existing data marketplace solutions [7, 22], data owners receive compensation for their data usages allocated by the broker. They have no means to set privacy preferences when supplying their data to the broker. The challenge to be addressed is: *How to formulate data owners' compensation functions based on not only the fair sharing of revenues allocated by the broker but also the data owners' requisites on privacy preservation*?

- *Model buyers.* As with the same practice of selling digital commodities in several versions, existing work [13] provides a set of models for sale with different levels of quality. However, their oversimplified noise injection-based version control quantifies the model quality via the magnitude of the noise, which does not directly align with model buyers' demands on Shapley coverage and resistance on model noise. The challenge is: *How to formulate model buyers' price functions based on not only resistance on model noise but also Shapley coverage*?

- *Broker.* Both data owners' compensation functions and model buyers' price functions should be taken into consideration by the broker when making market decisions. The challenge is: *How can the broker determine prices for a set of models to maximize the revenue with arbitrage-free guarantee, and train a set of models with maximum Shapley coverage given a manufacturing budget to remain competitive*?

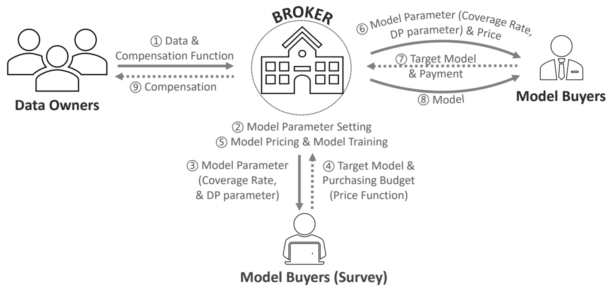

BROKER
① Data & Compensation Function
⑨ Compensation
Data Owners
⑥ Model Parameter (Coverage Rate, DP parameter) & Price
⑦ Target Model & Payment
⑧ Model
Model Buyers
② Model Parameter Setting
⑤ Model Pricing & Model Training
③ Model Parameter (Coverage Rate, & DP parameter)
④ Target Model & Purchasing Budget (Price Function)
Model Buyers (Survey)

Figure 1: An end-to-end data marketplace with model-based pricing. Please see Algorithm 8 for a complete pipeline.

Contributions. In this paper, we bridge the above gaps and address the identified challenges by proposing an en**D**-to-end mod**e**l m**a**rketp**l**ace with diff**e**rential p**r**ivacy (*Dealer*).

An illustration is provided in Figure 1, which includes three entities (i.e., data owners, broker, and model buyers) and their abbreviated interactions. From the data owners' perspective, *Dealer* proposes privacy sensitivity of data owners to model their requisites on privacy preservation and uses Shapley value to model fair sharing of revenues. Each data owner has a unique compensation function based on both privacy sensitivity and Shapley value [32]. From the model buyers' perspective, *Dealer* proposes Shapley coverage sensitivity and noise sensitivity of model buyers to model their demands on Shapley coverage and resistance on model noise. Each model buyer has a unique price function based on both Shapley coverage sensitivity and noise sensitivity. Both privacy sensitivity of data owners and noise sensitivity of model buyers are uniformly measured by the level of differential privacy (DP) [15, 16]. From the broker's perspective, *Dealer* depicts the full marketplace dynamics through two important functions including: 1) *model pricing* with the aim of maximizing revenue and at the same time following the market design principle of arbitrage-freeness, which prevents the model buyers from taking advantage of the marketplace by combining lower-tier models sold for lower prices into a higher-tier model to escape the designated price for that tier; and 2) *model training* with the aim of maximizing Shapley coverage given a manufacturing budget for each model version to remain competitive.

For model pricing, given a set of models with their Shapley coverage rates, DP parameters, and the pricing functions from model buyers, we design algorithms for the broker to find the optimal pricing to maximize the revenue with arbitrage-free guarantee. For model training, given the maximized revenue from model buyers or the manufacturing budget for each model, and the compensation functions of data owners and pricing functions of model buyers, we propose efficient algorithms for the broker to select a subset of data with the highest Shapley values to maximize the Shapley coverage in each model.

Our goal in this paper is not to provide a complete solution for the myriad problems involved in the model marketplace with differential privacy, but rather to introduce a framework with which to investigate some of the many questions and potentially open up a new research direction. We briefly summarize our contributions as follows.

- We present an end-to-end model marketplace with differential privacy *Dealer*, which is the first systematic study that includes all market participants. *Dealer* formalizes the abilities and restrictions of the three entities, formulates compensation functions for data owners and price functions for model buyers, and proposes the market decisions taken by the broker by formulating two optimization problems: revenue maximization problem with arbitrage-free constraint for model pricing and Shapley coverage maximization problem for model training.

- For the revenue maximization problem of model pricing, we show its complexity, discretize the search space, and propose an efficient dynamic programming algorithm. For the Shapley coverage maximization problem of model training, we show its complexity and propose efficient dynamic programming, greedy, and guess-based greedy algorithms with approximation guarantees.

- Experiments on the real chess dataset and synthetic datasets are conducted, which verify the efficiency and effectiveness of the proposed algorithms for model pricing and model training in *Dealer*.

Organization. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 provides the preliminaries, including the concept of Shapley value and its computation, versioning, arbitrage-free definition, and differential privacy related definitions and properties. We provide an end-to-end model marketplace with differential privacy in Section 4. Efficient algorithms for the optimization problems in *Dealer* are presented in Section 5. We report the experimental results and findings in Section 6. Finally, Section 7 draws a conclusion and discusses future work.

2. RELATED WORK

In this section, we discuss related work on compensation allocation and pricing mechanisms for data markets.

2.1 Compensation Allocation

In data marketplaces, a common way for compensation allocation is based on the importance of the data contributed by a data owner. An acquiescent method to evaluate importance of a data point to a model is leave-one-out (LOO) which compares the difference between the predictor's performance when trained on the entire dataset with and without the data point [10]. However, LOO does not satisfy the properties (balance, symmetry, zero element, and additivity) that we expect for the data valuation [32]. For example, given a point p in a dataset, if there is an exact copy p' in the dataset, removing p from this datasets does not change the predictor since p' is still there. Therefore, LOO will assign zero value to p regardless of how important p is, which violates the symmetry property.

Shapley value is a concept in cooperative game theory, which was named in honor of Lloyd Shapley. In contrast to LOO, Shapley value satisfies several desirable properties including balance, symmetry, zero element, and additivity [32]. Combined with its flexibility to support different utility functions, Shapley value has been extensively employed in the data pricing field [7, 8, 18, 22] to evaluate the data importance. One major challenge of applying Shapley value is its prohibitively high computational complexity. Evaluating the exact Shapley value involves the computation of the marginal utility of each user to every coalition of users, which is $\sharp P$-complete [17]. Such exponential computation is clearly impractical for evaluating a large number of training points. Even worse, for machine learning tasks, evaluating the utility function is extremely expensive due to model training. The worst case is that we need to train $O(2^n)$ models for computing the exact Shapley value for each data point. A number of approximation methods [8, 9, 17, 18] have been developed to overcome the computational hardness of the exact Shapley value. The most representative method is Monte Carlo method [9, 17], which is based on the random sampling of permutations.

In this paper, we allow data owners to set personalized compensation functions and the corresponding compensation is not only dependent on Shapley value but also based on privacy sensitivity.

2.2 Pricing Mechanisms

Envy-free query-based pricing. Ghosh et al. [19] initiated the study of markets for private data using differential privacy. They modeled the first framework in which data buyers would like to buy sensitive information to estimate a population statistic. They defined a property named envy-free for the first time. Envy-free ensures that no individual would prefer to switch their payment and privacy cost with each other. Guruswami et al. [20] studied the optimization problem of revenue maximization with envy-free guarantee. They investigated two cases of inputs: unit demand consumers and single minded consumers, and showed the optimization problem is APX-hard for both cases, which can be efficiently solved by a logarithmic approximation algorithm. Li et al. [27, 28, 29] presented the first theoretical framework for assigning value to noisy query answers as function of their accuracy, and for dividing the revenue among data owners who deserve compensation for their privacy loss. They defined an enhanced edition of envy-free, which is named arbitrage-free. Arbitrage-free pricing ensures the data buyer cannot purchase the desired information at a lower price by combing two low-price queries.

Arbitrage-free query-based pricing. Lin et al. [30] proposed necessary conditions for avoiding arbitrage and provide new arbitrage-free pricing functions. They also presented negative results related to the tension between flexible pricing and arbitrage-free, and illustrated how this tension often results in unreasonable prices. In addition to arbitrage-free, Koutris et al. [26] proposed another desirable property for the pricing function, discount-free, which requires that the prices offer no additional discounts than the ones specified by the broker. In fact, discount-free is the discrete version of arbitrage-free. Furthermore, they presented a polynomial time algorithm for pricing generalized chain queries. Recently, Chawla et al. [12] investigated three types of succinct pricing functions and studied the corresponding revenue maximization problems.

Arbitrage-free model-based pricing. Due to the increasing pervasiveness of machine learning based analytics, there is an emerging interest in studying the cost of acquiring data for machine learning. Chen et al. [13] proposed the first and the only existing model-based pricing framework which directly prices machine learning model instances with different noises rather than pricing the data. They formulated an optimization problem to find the arbitrage-free price that maximizes the revenue of the broker and proved such optimization problem is coNP-hard. However, 1) their work only focuses on the interactions between the broker and model buyers because the market research functionality of data owners can be completely replaced by the broker; 2) they assume there is only one (averaged) survey price point for each model, which is over-simplified; and 3) they do not have an explicit privacy analysis and guarantee for their proposed Gaussian mechanism that adds noise to the models.

Table 1: The summary of notations.

Notation	Definition
D_i	the i^{th} data owner
B_j	the j^{th} model buyer
M_k	the k^{th} model
$\mathcal{U}$	utility function
$\mathcal{SV}$	Shapley value
z_i	the i^{th} date set
ϵ	DP parameter
ρ_i	the privacy sensitivity of D_i
σ_j	the utility sensitivity of B_j
γ_j	the noise sensitivity of B_j
$\mathcal{CR}$	coverage rate function
$\mathcal{MB}$	manufacturing budget
$\langle p(\epsilon_1), ..., p(\epsilon_l)\rangle$	optimal pricing
$(\epsilon_k, sp^k[j])$	survey price point
$(\epsilon_k, p^k[j])$	complete price point

3. PRELIMINARIES

In this section, we introduce the preliminaries for our later development and summarize the frequently used notations in Table 1 for the convenience to readers.

3.1 Fairness and Shapley Value

Consider n data owners $D_1, \ldots, D_n$ such that data owner D_i owns data set z_i $(1 \leq i \leq n)$. We assume a utility function $\mathcal{U}(\mathcal{S})$ $(\mathcal{S} \subseteq \{z_1, \ldots, z_n\})$ that evaluates the utility of a coalition $\mathcal{S}$, which consists of datasets from multiple data owners, for a task, such as training a machine learning model. Shapley [32] lays out the fundamental requirements of fairness in market, including balance, symmetry, zero element, and additivity. Specifically, the Shapley value is a measure that can be used to evaluate data importance for allocation compensation while satisfying all the requirements. Shapley value measures the marginal utility improvement contributed by z_i

Algorithm 1: Monte Carlo Shapley value computation.

```
input  : data sets $z_1, \ldots, z_n$
output: Shapley value $\mathcal{SV}_i$ for each data set $z_i$ $(1 \le i \le n)$,
        and $\tau > 0$
1 initialize $\mathcal{SV}_i = 0$ $(1 \le i \le n)$;
2 for k=1 to $\tau$ do
3     let $\pi^k$ be a random permutation of $\{1, \ldots, n\}$;
4     for i=1 to n do
5         $\mathcal{SV}(z_{\pi^k(i)}) = \mathcal{U}(\{z_{\pi^k(1)}, \ldots, z_{\pi^k(i)}\}) - \mathcal{U}(\{z_{\pi^k(1)}, \ldots, z_{\pi^k(i-1)}\})$;
6         $\mathcal{SV}_{\pi^k(i)} += \mathcal{SV}(z_{\pi^k(i)})$;
7 return $\mathcal{SV}_i, \ldots, \mathcal{SV}_n$
```

from data owner D_i averaged over all possible coalitions of the data owners.

$$\mathcal{SV}_i = \frac{1}{n} \sum_{\mathcal{S} \subseteq \{z_1, \ldots, z_n\} \backslash z_i} \frac{\mathcal{U}(\mathcal{S} \cup \{z_i\}) - \mathcal{U}(\mathcal{S})}{\binom{n-1}{|\mathcal{S}|}} \tag{1}$$

Computing the exact Shapley value has to enumerate all the subsets and thus is prohibitively expensive. We adopt a commonly used Monte Carlo simulation method [9, 17] to compute the approximate Shapley value. We first sample random permutations of data sets from different data owners, and then scan each permutation from the first data set to the last one, and calculate the marginal contribution of every new data set. By examining a good number of permutations, the final estimation of Shapley value is simply the average of all the calculated marginal contributions. This Monte Carlo simulation gives an unbiased estimate of the Shapley value given enough number of permutations.

In practical applications, we can conduct Monte Carlo simulation iteratively until the average has empirically converged, as shown in Algorithm 1, where τ is the number of permutations. The larger the value of τ, the more accurate the computed Shapley value tends to be.

Based on the definition and the fair compensation property of Shapley value, we can easily generalize the Shapley value of one data set from one data owner to the Shapley value of a coalition of data sets from multiple owners.

Corollary 1. *Let $\mathcal{S}_D \subseteq \{D_1, \ldots, D_n\}$ be a subset of data owners. Then,*

$$\mathcal{SV}(\mathcal{S}_D) = \sum_{i: D_i \in \mathcal{S}_D} \mathcal{SV}_i$$

is a Shapley value that can be used for compensation allocation to the subset of data owners that satisfies all requirements of fairness.

3.2 Versioning

In a perfect world where the broker has infinite resource, the broker can sell a personalized model to each model buyer at an individualized price to maximize the revenue. However, such personalized pricing is rarely possible in practical applications. There is a practical and frequently adopted way, i.e., offering a small number of different versions of the product, which are designed to attract different types of buyers. Under this strategy, which is called versioning [31], buyers segment themselves. A version chosen by a buyer reveals the value that the buyer places on the data set and the price the buyer is willing to pay.

Following this industry practice, in our model marketplace, the broker trains l different model versions with different model privacy parameters which are measured by differential privacy.

3.3 Arbitrage-free Pricing

When multiple versions are available in a market, it is possible to derive a more expensive version from one or multiple versions with lower total cost. In such a scenario, arbitrage sneaks in. Arbitrage complicates interactions between the broker and buyers. Buyers have to carefully choose versions to achieve the lowest price, while the broker may not maximize the revenue intended by the released prices. Therefore, arbitrage-free pricing functions are highly desirable. A pricing function is arbitrage-free if it satisfies the following two properties [13, 29]. It prevents model buyers from taking advantage of the marketplace by combining lower-tier versions sold for cheaper prices into a higher-tier version to escape the designated price for that tier.

Property 1. ***(Monotonicity).*** *Given a function $f : (\mathbb{R}^+)^n \to \mathbb{R}^+$, f is monotone if and only if for any two vectors $\mathbf{x}, \mathbf{y} \in (\mathbb{R}^+)^n$, $\mathbf{x} \le \mathbf{y}$, $f(\mathbf{x}) \le f(\mathbf{y})$.*

Property 2. ***(Subadditivity).*** *Given a function $f : (\mathbb{R}^+)^n \to \mathbb{R}^+$, f is subadditive if and only if for any two vectors $\mathbf{x}, \mathbf{y} \in (\mathbb{R}^+)^n$, $f(\mathbf{x} + \mathbf{y}) \le f(\mathbf{x}) + f(\mathbf{y})$.*

3.4 Differential Privacy

Differential privacy [15, 16] is a formal mathematical framework rigorously providing privacy protection. To the best of our knowledge, none of the existing model marketplaces adopt or consider differential privacy.

Definition 1. *(Differential Privacy) A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private, if for any pair of datasets $\mathcal{S}$ and $\mathcal{S}'$ that differs in one data sample, and for all possible output $\mathcal{OUT}$ of $\mathcal{A}$, the following holds,*

$$\mathbb{P}[\mathcal{A}(\mathcal{S}) \in \mathcal{OUT}] \le e^{\epsilon} \mathbb{P}[\mathcal{A}(\mathcal{S}') \in \mathcal{OUT}] + \delta, \tag{2}$$

where the probability is taken over the randomness of $\mathcal{A}$.

In short, this formulation implies that the output of a differentially private algorithm is indistinguishable between two datasets that differ in one sample. ϵ captures the degree of indistinguishability, which intuitively can be seen as the upper bound of the privacy loss for each data sample. The smaller ϵ is, the more indistinguishable the outputs are, and the less the privacy loss is. The other parameter δ captures the probability that the privacy loss is out of the upper bound. Given a specified ϵ, the closer δ is to 0, the less the probability of privacy breach is, and the better the algorithm is. In practice, for a meaningful DP guarantee, the parameters are chosen as $0 < \epsilon \le 10$, $\delta \ll \frac{1}{n}$, where n is the number of data owners [21]. In this paper, the broker will train a set of models with differential privacy guarantees. According to the robustness of post-processing of differential privacy, any post-processing of the models by the model buyers will not incur additional privacy loss. For simplicity, we use a uniform small value δ and different ϵ for different versions of models.

Lemma 1. *(Simple Composition [16]) Considering that there are J randomized algorithms and each of them is (ϵ_j, δ_j)-differentially private for $j \in \{1, \ldots, J\}$, then sequentially applying these J algorithms satisfies $(\sum_{j=1}^{J} \epsilon_j, \sum_{j=1}^{J} \delta_j)$-differential privacy.*

Lemma 1 is essential for DP mechanism design and analysis, which enables algorithm designers to compose elementary DP operations into a more sophisticated one. Importantly, we will show later that it plays a crucial role in model market design as well. Lemma 1

Algorithm 2: Objective perturbation for differentially private model training.

```
input  : $Z_{train}$ and $(\epsilon, \delta)$.
output: $w_{DP}$.
1 Sample $N_1 \sim \mathcal{N}(\mathbf{0}_d, \sigma_1^2 I_d)$, where $\sigma_1 = \frac{20L^2 \log(1/\delta)}{\epsilon^2}$ and $L$ is a
  $L_2$-Lipschitz constant [21];
2 Objective Perturbation:
  $\mathcal{L}_{OP}(w) = \mathcal{L}(w; Z_{train}) + \lambda\|w\|_2^2 + \frac{1}{n}\langle N_1, w\rangle$;
3 Optimize $\mathcal{L}_{OP}(w)$ to obtain $\alpha$ approximate solution $\hat{w}$;
4 Sample $N_2 \sim \mathcal{N}(\mathbf{0}_d, \sigma_2^2 I_d)$, where $\sigma_2 = \frac{40\alpha \log(1/\delta)}{\lambda\epsilon^2}$;
5 return $w_{DP} = proj_{\Omega}(\hat{w} + N_2)$;
```

suggests that differential privacy is an appropriate mechanism for versioning models with arbitrage-free pricing. The challenge is how to ensure the pricing of the versioned models is arbitrage-free.

To train a model satisfying differential privacy, a popular method is objective perturbation [11], which perturbs the objective function of the model by quantified noise. For the perturbed objective, we adopts gradient descent-based iterative optimization algorithms to obtain an approximate solution, which is popular in practical machine learning model training. However, as pointed out in [21], conventional objective perturbation establishes DP guarantee for the exact optimum of the perturbed objective function, while the DP guarantee is unknown for the approximate solution. In order to allow the broker to utilize the practical iterative training algorithms and still achieve differential privacy, in this paper, we follow the enhanced objective perturbation called *approximate minima perturbation*, which allows solving the perturbed objective up to α approximation. It uses a two-phase noise injection strategy that perturbs both the objective and the approximate output. The idea is summarized in Algorithm 2. In particular, Line 2 perturbs the model with calibrated noise N_1, Line 3 optimizes the perturbated model followed by an output perturbation with noise N_2 in Line 4. Finally, w_{DP} is obtained by a projection to the constrained set Ω. As shown by the next lemma, Algorithm 2 trains an (ϵ, δ)-DP model based on training dataset Z_{train}, which outputs model parameter w_{DP}.

Lemma 2. *(Theorem 1 in [21]) Algorithm 2 is (ϵ, δ)-differentially private.*

In *Dealer*, the broker will train a series differentially private models by invoking Algorithm 2.

4. DEALER: STRUCTURE AND PARTIES

In this section, we propose a model marketplace *Dealer*. We describe the structure of *Dealer* including the parties and their operations in the marketplace. We first give an overview of *Dealer* and then specify the roles and mechanisms in detail.

4.1 Overview

The objective of *Dealer* is to bridge the supplies from data owners and the demands from model buyers. We assume one broker who collects data from multiple data owners, designs and builds models, and sells the models to multiple model buyers. Data owners, broker, and model buyers have different concerns and requirements.

Data owners have two concerns. First, they are concerned about privacy protection. The less privacy protection in models built, the more compensation data owners ask for. Moreover, data owners expect fair sharing of revenues from models sold to model buyers. In *Dealer*, we propose *privacy sensitivity* of data owners to model their requisites on privacy preservation and use Shapley values to model fair sharing of revenues.

Model buyers are concerned about the coverage of the data that are used to build a model captured by the Shapley value of the data and the noise added to achieve privacy preservation. The more coverage of the data and the less noise added, the higher the value a model to buyers. In *Dealer*, we propose *Shapley coverage sensitivity* and *noise sensitivity* of model buyers to model their demands on Shapley coverage and resistance to noise, respectively. To capture the economic constraints, we assume that each model buyer has a budget. The broker can accordingly calculate the Shapley coverage of a given model, which is captured by the Shapley value of the selected data used to train the model. We assume that model buyers always buy the cheapest model satisfying their demands.

The broker defines and builds models using data collected from data owners and sells the models to model buyers. Without loss of generality, we assume that the broker is neutral and does not charge any cost in model building, i.e., the total revenue from model sales is fully and fairly distributed to data owners. To practice *Dealer*, the broker can easily factor in a percentage of total revenue for model building and other cost, which does not affect our mechanisms. The objective of the broker is to maximize the total revenue.

To keep our discussion simple, we assume that all parties in *Dealer*, including data owners, broker, and model buyers, are honest. That is, no cheating activities exist in the marketplace.

4.2 Data Owners

We assume n data owners $D_1, \ldots, D_n$. For the sake of clarity, we overload the symbol D_i $(1 \leq i \leq n)$ to also denote the data owned by D_i. Each data owner D_i $(1 \leq i \leq n)$ is willing to share her data with the broker for compensation. The compensation requested depends on the data owner's privacy disclosed through the models sold by the broker. Moreover, D_i expects the fair sharing of the revenues with other data owners based on the utility of the data contributed.

In this paper, we use ϵ-differential privacy to quantify possible user privacy disclosure. Technically, we use a monotonic *compensation function* c_i for data owner D_i to model the data owner's requested compensation for a model that uses data D_i and satisfies ϵ-differential privacy.

$$c_i(\epsilon) = b_i \cdot s_i(\epsilon) \tag{3}$$

where $\epsilon \geq 0$ is the differential privacy parameter, and b_i is the base price and should be proportional to the Shapley value of D_i with regard to all data sets $\{D_1, \ldots, D_n\}$ by other peers contributed to building models. Function $s_i(\epsilon)$ reflects the user's belief of privacy in price. The larger the value of ϵ, the less protection provided by ϵ-differential privacy, and thus the higher the compensation requested. In general, any monotonic function with respect to ϵ can be used. Notice that factor e^ϵ in the definition of ϵ-differential privacy controls the tolerance of privacy disclosure. The larger the value of e^ϵ, the less privacy protection. In this paper, we use the following function for $s_i(\epsilon)$ in our discussion.

$$s_i(\epsilon) = (e^\epsilon)^{\rho_i} = e^{\rho_i \cdot \epsilon} \tag{4}$$

Parameter $\rho_i \geq 0$ tunes the marginal increase of the user's price elasticity of privacy, and is called the user's *privacy sensitivity*. When $0 \leq \rho_i < 1$, $s_i(\epsilon)$ grows sub-linearly with respect to e^ϵ; when $\rho_i > 1$, the growth is super-linear; and when $\rho_i = 1$, the growth is the same as e^ϵ. A very privacy-conservative data owner can have a super-linear function, which means a higher compensation if her data is used, but may not get selected for model building because the broker has limited manufacturing budget.

4.3 Model Buyers

We assume m model buyers $B_1, \ldots, B_m$, each carrying budgets $V_1, \ldots, V_m$, respectively, where $V_j > 0$ ($1 \leq j \leq m$). In general, the value of the models depend on both the coverage of the data being used for building the model and the noise added to the model for privacy protection.

First, buyers' offer prices are sensitive to the coverage of the data used. The ideal model takes all data sets from all data owners $D_1 \ldots, D_n$. Let the total coverage be C_{all}. A model buyer may set an expectation requirement on the percentage of the Shapley coverage that should be used to build the model, and the price that the buyer is willing to pay may change when the Shapley coverage changes. When using different subsets of data to build a model, not only the quantity, but also the coverage of the data matters for the final accuracy of the model. Naturally, we can use Shapley value to measure the Shapley coverage.

Formally, for a model M that is built using k data sets $D_{i_1}, \ldots, D_{i_k}$ from their data owners, the *coverage rate* of the model M is

$$\mathcal{CR}(M) = \mathcal{CR}(D_{i_1}, \ldots, D_{i_k}) = \frac{\mathcal{SV}(\{D_{i_1}, \ldots, D_{i_k}\})}{\mathcal{SV}(\{D_1, \ldots, D_n\})} \tag{5}$$

Technically, a model buyer B_j ($1 \leq j \leq m$) can set a *coverage expectation* θ_j ($0 < \theta_j \leq 1$). The price B_j is willing to pay for a model M with respect to coverage rate $CR(M)$ can be

$$Price_{coverage}(B_j) = V_j \cdot \frac{1}{1 + e^{-\delta_j (CR(M) - \theta_j)}} \tag{6}$$

where parameter $\delta_j > 0$ is B_j's *coverage sensitivity*, which controls how quickly the buyer loses interest in the model if the model does not meet the Shapley coverage expectation θ_j. The larger the value of δ_j, the sharper the change of the price at the expectation point θ_j.

Second, buyers' offer prices are also sensitive to noise added to the model. The ideal model does not add any noise. The effect of noises added for privacy protection can be measured quantitatively by the parameter ϵ in the ϵ-differential privacy mechanism. In ϵ-differential privacy, no noise is added when $\epsilon = \infty$. Technically, a model buyer B_j ($1 \leq j \leq m$) can set a *noise expectation* η_j ($\eta_j > 0$) and a parameter γ_j ($\gamma_j > 0$), called *noise sensitivity*, to express the buyer's price sensitivity with respect to noise: the larger the value of γ_j, the more sensitive the buyer is with respect to noise.

Putting the above two aspects together, for a buyer B_j ($1 \leq j \leq m$), let V_j be the buyer's total budget to purchase a model, θ_j the expectation on Shapley coverage in Shapley value, δ_j the Shapley coverage sensitivity, η_j the expectation on model noise in differential privacy, and γ_j the noise sensitivity. Then, for a model M that is built using data sets $D_{i_1}, \ldots, D_{i_k}$ from data owners and satisfies ϵ-differential privacy, we use the following *price function* for model buyer B_j on model M

$$P(B_j, M) = V_j \cdot \frac{1}{1 + e^{-\delta_j (CR(M) - \theta_j)}} \cdot \frac{1}{1 + e^{-\gamma_j (\epsilon - \eta_j)}} \tag{7}$$

Clearly, $P(B_j, M)$ reflects model buyer B_j's assessment of the value of model M, thus is also called the *model value* of M for B_j.

4.4 Broker

The broker collects data from data owners, builds and sells models to model buyers. To accommodate different data owners' requirements on privacy preservation and various model buyers' demands on Shapley coverage and noise sensitivity, multiple versions of a model may be developed. In practice, the number of versions made available to model buyers is often kept small. We also call a version of a model simply a model.

Assume that the broker builds l versions of a model, $M_1, \ldots, M_l$. A model M_k ($1 \leq k \leq l$) is specified as a tuple $M_k = (\mu_k, \epsilon_k, p_k)$, where $\mu = CR(M_k)$ is the coverage rate of the model, ϵ_k is the level of differential privacy guarantee, and p_k is the price set by the broker. To make the marketplace concise, we list the models in μ value and ϵ value ascending order, that is, $\mu_{k_1} \leq \mu_{k_2}$ and $\epsilon_{k_1} \leq \epsilon_{k_2}$ if $k_1 \leq k_2$. Let $r(M_k, D_i)$ be the compensation allocated to data owner D_i who contributes the data to model M_k. The compensation is decided by the broker.

In model marketplace *Dealer*, we make the following assumptions about the participants' behavior.

- (Participation) A data owner D_i contributes the data to a model M_k as long as the broker can allocate a payment no less than the data owner asks for, i.e., $r(M_k, D_i) \geq c_i(\epsilon_k)$. A data owner participates in and collects compensation from every model that meets the requirements.

- (Fairness) For any model M_k and data owners D_{i_1} and D_{i_2} who contribute their data to the model,

  $$\frac{r(M_k, D_{i_1})}{c_{i_1}(\epsilon_k)} = \frac{r(M_k, D_{i_2})}{c_{i_2}(\epsilon_k)}$$

  that is, the compensation among all contributing data owners within each model is fair.

- (Cost minimization purchase) A model buyer B_j buys one and only one model as long as there exists a model M_k such that

  $$\theta_j \leq \mu_k \text{ and } \eta_j \leq \epsilon_k \tag{8}$$

  If there is no model M_k satisfying Eq. 8, then B_j purchases nothing; if there is only one model M_k satisfying Eq. 8, then B_j purchases M_k with budget $P(B_j, M_k)$; if there are multiple models $M_{k_1}, \ldots, M_{k_{l'}}$ satisfying Eq. 8, then B_j purchases model with $\arg\min_{k \in \{k_1, \ldots, k_{l'}\}} \epsilon_k$, i.e., the model having the lowest ϵ with the lowest price for B_j, with budget $P(B_j, M_{\arg\min_{k \in \{k_1, \ldots, k_{l'}\}} \epsilon_k})$.

- (Neutral broker) For each model M_k that uses data sets $D_{i_1}, \ldots, D_{i_{n'}}$ and is purchased by model buyers $B_{j_1}, \ldots, B_{j_{m'}}$, $\sum_{i \in \{i_1, \ldots, i_{n'}\}} r(M_k, D_i) = m' \cdot p_k$, i.e., the revenue produced by each model is fully distributed to all data contributors. In this paper, we assume a neutral broker to keep our discussion simple. In practice, a broker can easily factor in a percentage of total revenue for model building and other cost, which does not affect our mechanisms and algorithms.

The broker is responsible for defining versions of model. Ideally, the arbitrage-free guarantee may be applied to both μ_k and ϵ_k. However, the arbitrage-free guarantee is only applied to one variable in the existing works [13, 27, 28, 29]. It is not clear or feasible how we can apply the arbitrage-free guarantee to two variables. Because ϵ_k has much greater impact than μ_k on model accuracy (Figure 3), in this paper, the broker ensures *arbitrage-free* guarantee with respect to ϵ_k. Now we are ready to state the optimization problem in *Dealer*. Given a set of data owners, a set of model buyers, and the number of models l, the broker needs to define l models $M_1, \ldots, M_l$ such that the total revenue $\sum_{k=1}^{l} p_k \cdot purchase(p_k)$ is maximized, where $purchase(p_k)$ is the number of model buyers purchasing M_k. In the rest of the paper, we refer to this problem as *revenue maximization problem.*

5. DEALER: MODEL PRICING AND MODEL TRAINING

In this section, we study the revenue maximization problem for model pricing in Section 5.1. Given the maximized revenue from model buyers, the broker needs to maximize the Shapley coverage for each model version to remain competitive and see if the Shapley coverage satisfies the predefined Shapley coverage. We study the Shapley coverage maximization problem for model training in Section 5.2. We also summarize the complete *Dealer* dynamics in Section 5.3.

5.1 Revenue Maximization Problem for Model Pricing

Prior to releasing models and setting the prices for sale, the broker needs to conduct a market survey to collect each model buyer's price function $P(B_j, M_k)$ in order to price the models to maximize the revenue, which can be done by the broker herself or third-party companies. Let the survey size be m', i.e., m' potential model buyers are recruited to provide their price function $P(B_j, M_k)$. Given a set of models the broker plan to train with coverage rate μ_k and DP parameter ϵ_k, for the j^{th} survey participant B_j, the broker calculates (tm_j, v_j) according to Eq. 8 if there is at least one model satisfying B_j's requirements on both Shapley coverage and model noise, where tm_j indicates the target model of B_j and v_j is the budget by the buyer for purchasing tm_j. We call each (tm_j, v_j) a *survey price point* in the following.

With the m' tuples computed from the surveyed buyers, the broker will price each model in the aim of maximizing revenue and at the same time following the market design principle of arbitrage-free pricing. The revenue maximization ($\mathcal{RM}$) problem is formulated as follows.

$$\arg\max_{\langle p(\epsilon_1),\ldots,p(\epsilon_l)\rangle} \sum_{k=1}^{l}\sum_{j=1}^{m'} p(\epsilon_k)\cdot \mathbb{I}(tm_j == M_k)\cdot \mathbb{I}(p(\epsilon_k)\le v_j), \quad (9)$$

$$s.t.\ p(\epsilon_{k_1}+\epsilon_{k_2}) \le p(\epsilon_{k_1}) + p(\epsilon_{k_2}),\ \epsilon_{k_1}, \epsilon_{k_2} > 0, \quad (10)$$

$$0 < p(\epsilon_{k_1}) \le p(\epsilon_{k_2}),\ 0 < \epsilon_{k_1} \le \epsilon_{k_2}, \quad (11)$$

where $\mathbb{I}(tm_j == M_k)$ indicates whether B_j's target model is M_k, $\mathbb{I}(p(\epsilon_k) \le v_j)$ indicates whether the price for model M_k (p_k or $p(\epsilon_k)$ from the DP parameter perspective) with DP parameter ϵ_k is less than or equal to the budget of B_j for purchasing M_k. The optimal solution of this revenue maximization problem has the arbitrage-free guarantee since Eq. 10 ensures subadditivity (Property 2) and Eq. 11 ensures monotonicity (Property 1).

We refer to the maximum revenue for $\mathcal{RM}$ as $MAX(\mathcal{RM})$ and use $(\epsilon_k, sp^k[j])$ to denote the j^{th} lowest survey price point in model M_k with DP parameter ϵ_k. For example, we assume $\epsilon_1 = 1$, $\epsilon_2 = 2$, and $\epsilon_3 = 3$ in Figure 2. We have six survey points (participants) shown in black disk $(1, sp^1[1] = 1)$, $(1, sp^1[2] = 4)$, $(2, sp^2[1] = 3)$, $(2, sp^2[2] = 7)$, $(3, sp^3[1] = 5)$, and $(3, sp^3[2] = 8)$, where $(2, sp^2[1] = 3)$ means one model buyer would like to purchase model M_2 with DP parameter ϵ_2 using budget 3. These survey price points make up a *survey price space*.

A special case of the $\mathcal{RM}$ problem has been studied earlier [13] given one (averaged) survey price point for each model. Given one survey price point $(\epsilon_k, sp^k[1])$ for each model M_k, $k = 1, ..., l$, determining whether there exists a pricing function $p(\epsilon_k)$ that 1) is positive, monotone, and subadditive; and 2) ensures $p(\epsilon_k) = sp^k[1]$ for all $k = 1, ..., l$, has been shown as a co-NP hard problem. It is easy to see that this co-NP hard problem is a special case of the $\mathcal{RM}$ problem which has multiple survey price points for each model.

In order to overcome the hardness of the original optimization $\mathcal{RM}$ problem, we seek to approximately solve the problem by relaxing the subadditivity constraint. We relax the constraint of $p(\epsilon_{k_1} + \epsilon_{k_2}) \le p(\epsilon_{k_1}) + p(\epsilon_{k_2})$ in Eq. 10 to

$$p(\epsilon_{k_1})/\epsilon_{k_1} \ge p(\epsilon_{k_2})/\epsilon_{k_2}, \epsilon_{k_1} \le \epsilon_{k_2}, \quad (12)$$

which still satisfies the requirement of arbitrage-free pricing because this new constraint is sufficient but not necessary for the subadditivity constraint. We refer to this relaxed problem as relaxed revenue maximization ($\mathcal{RRM}$) problem. Intuitively, we want to make sure that the unit price for large purchases is smaller than or equal to the unit price for small purchases.

In the following, we show the maximum revenue for $\mathcal{RRM}$, $MAX(\mathcal{RRM})$, has a lower bound with respect to the maximum revenue for $\mathcal{RM}$, $MAX(\mathcal{RM})$.

Theorem 1. *The maximum revenue for $\mathcal{RM}$ has the following relationship with the maximum revenue for $\mathcal{RRM}$,*

$$MAX(\mathcal{RRM}) \ge MAX(\mathcal{RM})/2.$$

Proof. Given a feasible solution $p_{\mathcal{RM}}$ of the revenue maximization problem, we construct a solution $p_{\mathcal{RRM}}$ such that for all $k_1 > 0$, $p_{\mathcal{RRM}}(\epsilon_{k_1}) = \epsilon_{k_1} \times min_{0<x\le k_1}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\}$, where $p_{\mathcal{RRM}}(\epsilon_{k_1})$ is the price in $\mathcal{RRM}$ for model M_{k_1} with model DP parameter ϵ_{k_1}. Let $0 < k_1 \le k_2$. We show that $p_{\mathcal{RRM}}$ is a feasible solution of the relaxed maximization problem as follows.

We first prove that $p_{\mathcal{RRM}}$ satisfies monotonicity. Let $x'_{min} = \arg min_{0<x\le k_2}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\}$. We have two cases for x'_{min}, $0 < x'_{min} \le k_1$ and $k_1 < x'_{min} \le k_2$. For the first case $0 < x'_{min} \le k_1$, we have $min_{0<x\le k_2}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\} = min_{0<x\le k_1}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\}$ because x'_{min} lies in the range of $(0, k_1]$. And then we have $p_{\mathcal{RRM}}(\epsilon_{k_2}) = \epsilon_{k_2} \times min_{0<x\le k_2}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\} \ge \epsilon_{k_1} \times min_{0<x\le k_1}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\} = p_{\mathcal{RRM}}(\epsilon_{k_1})$. For the second case $k_1 < x'_{min} \le k_2$, we have $p_{\mathcal{RRM}}(\epsilon_{k_1}) = \epsilon_{k_1} \times min_{0<x\le k_1}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\} \le \epsilon_{k_1} \times \{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\}_{x=k_1} = \epsilon_{k_1} \times \{p_{\mathcal{RM}}(\epsilon_{k_1})/\epsilon_{k_1}\} = p_{\mathcal{RM}}(\epsilon_{k_1}) < p_{\mathcal{RM}}(\epsilon_{x'_{min}}) = \epsilon_{x'_{min}}\{p_{\mathcal{RM}}(\epsilon_{x'_{min}})/\epsilon_{x'_{min}}\}$. Because $x'_{min} \le k_2$ and $p_{\mathcal{RM}}(\epsilon_{x'_{min}})/\epsilon_{x'_{min}} = min_{0<x\le k_2}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\}$, we have $\epsilon_{x'_{min}}\{p_{\mathcal{RM}}(\epsilon_{x'_{min}})/\epsilon_{x'_{min}}\} \le \epsilon_{k_2} \times min_{0<x\le k_2}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\} = p_{\mathcal{RRM}}(\epsilon_{k_2})$. That is $p_{\mathcal{RRM}}(\epsilon_{k_1}) < p_{\mathcal{RRM}}(\epsilon_{k_2})$.

We then prove that $p_{\mathcal{RRM}}$ satisfies subadditivity. We have $p_{\mathcal{RRM}}(\epsilon_{k_1})/\epsilon_{k_1} = min_{0<x\le k_1}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\} \ge min_{0<x\le k_2}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\} = p_{\mathcal{RRM}}(\epsilon_{k_2})/\epsilon_{k_2}$. Therefore, we have $p_{\mathcal{RRM}}(\epsilon_{k_1})/\epsilon_{k_1} \ge p_{\mathcal{RRM}}(\epsilon_{k_2})/\epsilon_{k_2}$ which is the subadditivity constraint.

Now we prove the lower bound. Let $x_{min} = \arg min_{0<x\le k}\{p_{\mathcal{RM}}(\epsilon_x)/\epsilon_x\}$, i.e., $x_{min} \le k$. We show that for every $k > 0$, we have $p_{\mathcal{RM}}(\epsilon_k)/2 \le p_{\mathcal{RRM}}(\epsilon_k)$ as follows. We have $p_{\mathcal{RM}}(\epsilon_k) = p_{\mathcal{RM}}(\epsilon_{x_{min}}\frac{\epsilon_k}{\epsilon_{x_{min}}}) \le p_{\mathcal{RM}}(\epsilon_{x_{min}}\lceil\frac{\epsilon_k}{\epsilon_{x_{min}}}\rceil) \le \lceil\frac{\epsilon_k}{\epsilon_{x_{min}}}\rceil p_{\mathcal{RM}}(\epsilon_{x_{min}})$ because $p_{\mathcal{RM}}$ satisfies the subadditivity constraint. Therefore, we have $p_{\mathcal{RRM}}(\epsilon_k) = \epsilon_k\{\frac{p_{\mathcal{RM}}(\epsilon_{x_{min}})}{\epsilon_{x_{min}}}\} \ge \frac{\epsilon_k}{\epsilon_{x_{min}}}\{\frac{p_{\mathcal{RM}}(\epsilon_k)}{\lceil\frac{\epsilon_k}{\epsilon_{x_{min}}}\rceil}\} \ge \frac{\epsilon_k}{\epsilon_{x_{min}}}\{\frac{p_{\mathcal{RM}}(\epsilon_k)}{\frac{\epsilon_k}{\epsilon_{x_{min}}}+1}\} \ge p_{\mathcal{RM}}(\epsilon_k)/2$ because $x_{min} \le k$. Because $p_{\mathcal{RRM}}(\epsilon_k) = \epsilon_k \times min_{0<x\le k}\{\frac{p_{\mathcal{RM}}(\epsilon_x)}{\epsilon_x}\} \le \epsilon_k \times \{\frac{p_{\mathcal{RM}}(\epsilon_x)}{\epsilon_x}\}_{x=k} = p_{\mathcal{RM}}(\epsilon_k)$, we have $p_{\mathcal{RRM}}(\epsilon_k) \le p_{\mathcal{RM}}(\epsilon_k)$. Therefore, for each $k > 0$, we have $\sum_{j=1}^{m'} \cdot\mathbb{I}(tm_j == M_k)\cdot\mathbb{I}(p_{\mathcal{RRM}}(\epsilon_k) \le v_j) \ge \sum_{j=1}^{m'} \cdot\mathbb{I}(tm_j == M_k)\cdot\mathbb{I}(p_{\mathcal{RM}}(\epsilon_k) \le v_j)$. With $p_{\mathcal{RM}}(\epsilon_k)/2 \le p_{\mathcal{RRM}}(\epsilon_k)$, we conclude that $MAX(\mathcal{RM})/2 \le MAX(\mathcal{RRM})$. □

Dynamic Programming Algorithm. We show an efficient dynamic programming algorithm to solve the relaxed revenue maximization problem.

At first glance, for each model, it seems that all possible values in the price range can be an optimal price, which makes the problem arguably intractable to solve. In the following, we show how to

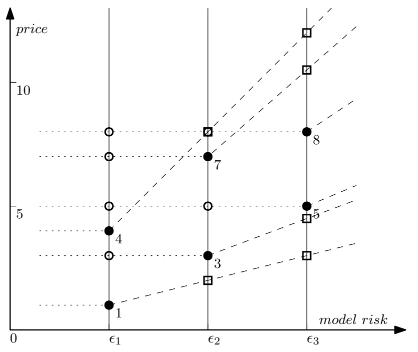


Figure 2: Revenue maximization example.

construct a *complete price space* in the discrete space and prove the complete price space is sufficient to obtain the maximum revenue.

Constructing Complete Price Space. It is easy to see that those survey price points could be potential solutions. For each survey price point $(\epsilon_k, sp^k[j])$, it determines unit price $sp^k[j]/\epsilon_k$ and price $sp^k[j]$. The general idea is that if we choose $(\epsilon_k, sp^k[j])$ as the optimal price point in model M_k, it affects the price for models $M_{k'}, k' = 1, ..., k-1$ due to the monotonicity constraint and the unit price for models $M_{k'}, k' = k+1, ..., l$ due to the subadditivity constraint. If we set the optimal price in model M_k as $sp^k[j]$, the unit price of the following models after model M_k cannot be larger than $sp^k[j]/\epsilon_k$. Therefore, for each survey price point $(\epsilon_k, sp^k[j])$, we draw one line $l_{(\epsilon_k, sp^k[j])}$ through survey price point $(\epsilon_k, sp^k[j])$ and the original point. For each model $M_{k'}$ with DP parameter $\epsilon_{k'}$, we draw one vertical line $l_{\epsilon_{k'}}$ through $(\epsilon_{k'}, 0)$. By intersecting line $l_{(\epsilon_k, sp^k[j])}$ and line $l_{\epsilon_{k'}}$, we obtain $l-k$ new price points $(\epsilon_{k'}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k'})$ for $k' = k+1, ..., l$. We note that we do not need to generate the price points as candidates for $k' = 1, ..., k-1$ because the unit price of model M_k can only constrain the unit price of model $M_{k'}, k' = k+1, ..., l$ (due to the subadditivity constraint). Furthermore, for each model, its price is also determined by the survey price points of its right neighbors (due to the monotonicity constraint). Therefore, we need to add the survey price points of model M_k to models $M_{k'}, k' = 1, ..., k-1$. The detailed algorithm for constructing the complete price space is shown in Algorithm 3. In Lines 11-15, we use $f(\epsilon_k, p^k[j])$ to distinguish the survey price points from the other points in the complete price space. For ease of presentation, we name the price point in the complete price space from Line 1 as $\mathcal{SU}$ (survey) point, the price point from Line 8 as $\mathcal{SC}$ (subadditivity constraint) point, and the price point from Line 10 as $\mathcal{MC}$ (monotonicity constraint) point.

Example 1. *We show a running example of Algorithm 3. In Figure 2, we add the survey price points* $(1, sp^1[1] = 1)$, $(1, sp^1[2] = 4)$, $(2, sp^2[1] = 3)$, $(2, sp^2[2] = 7)$, $(3, sp^3[1] = 5)$, *and* $(3, sp^3[2] = 8)$ *to the complete price space in Line 1. In Line 2, for the survey price point* $(1, 1)$, *we draw a line* $l_{(1,1)}$ *through this point and the original point in Line 3. In Line 4, for model* M_1 *with privacy parameter* $\epsilon_1 = 1$, *we draw a vertical line* l_{ϵ_1}. *In Lines 6-8, for* $l_{(1,1)}$ *and* l_{ϵ_2}, *we add intersection point* $(\epsilon_2, \frac{sp^1[1]}{\epsilon_1}\epsilon_2) = (2, 2)$ *to the complete price space. In total, we have six such new price points shown in box. In Line 9, for survey price point* $(3, sp^3[1]) = (3, 5)$, *we add price points* $(2, 5)$ *and* $(1, 5)$ *to the complete price space in Line 10. Similarly, we also have six such new price points shown in circle. Therefore, for the complete price space, we have six price points for models* M_1 *and* M_3. *We have five price points for model* M_2 *because the intersection point of* $(\epsilon_2, \frac{sp^1[2]}{\epsilon_1}\epsilon_2) = (2, 8)$ *is same as the* $\mathcal{MC}$ *point* $(\epsilon_k, sp^3[2]) = (2, 8)$ *for* $k = 2$. *In Lines 12-15, we have* $f(2, 3) = 1$ *and* $f(2, 5) = 0$.

Algorithm 3: Constructing complete price space for the relaxed revenue maximization problem.

```
input : Model with noise parameter $\epsilon_k$ and the $j^{th}$ lowest
        survey price point for model with noise parameter
        $\epsilon_k$, denoted as $(\epsilon_k, sp^k[j])$.
output: Complete price space.
1  add all the survey price points $(\epsilon_k, sp^k[j])$ to the complete
     price space;
2  for each survey price point $(\epsilon_k, sp^k[j])$ do
3      draw a line $l_{(\epsilon_k, sp^k[j])}$ through this point and the original
         point;
4  for each model with noise parameter $\epsilon_k$ do
5      draw a vertical line $l_{\epsilon_k}$;
6  for each line $l_{\epsilon_{k'}}$ do
7      for each line $l_{(\epsilon_k, sp^k[j])}$ do
8          add point $(\epsilon_{k'}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k'})$ by intersecting line $l_{\epsilon_{k'}}$ and
             line $l_{(\epsilon_k, sp^k[j])}$ to the complete price space for
             $k' = k+1, ..., l$;
9  for each survey price point $(\epsilon_k, sp^k[j])$ do
10     add price point $(\epsilon_{k'}, sp^k[j])$ to the complete price space
         for $k' = 1, ..., k-1$;
11 for each price point $(\epsilon_k, p^k[j])$ in the complete price space
     do
12     if $(\epsilon_k, p^k[j'])$ is a survey price point then
13         $f(\epsilon_k, p^k[j]) = 1$;
14     else
15         $f(\epsilon_k, p^k[j]) = 0$;
```

Theorem 2. *The complete price space constructed by Algorithm 3 is sufficient for finding the optimal solution of the relaxed revenue maximization problem.*

Proof. As we discussed in constructing complete price space, for each survey price point $(\epsilon_k, sp^k[j])$, it can affect the model revenue of model M_k, the unit price for models $M_{k'}, k' = k+1, ..., l$ and the price for models $M_{k'}, k' = 1, ..., k-1$. We prove that the $\mathcal{SC}$ and $\mathcal{MC}$ points are non-recursive, i.e., we do not need to generate new price points in the complete price space based on the generated $\mathcal{SC}$ and $\mathcal{MC}$ points.

Given a survey price point $(\epsilon_k, sp^k[j])$ in model M_k, it determines a $\mathcal{SC}$ point in model M_{k_l}, $(\epsilon_{k_l}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l})$ for $k_l > k$. We do not need to generate $\mathcal{SC}$ points based on $(\epsilon_{k_l}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l})$ because it has the same $\mathcal{SC}$ points for models $M_{k'}, k' = k_l+1, ..., l$ with $(\epsilon_k, sp^k[j])$. We may use $(\epsilon_{k_l}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l})$ to generate a $\mathcal{MC}$ point $(\epsilon_{k'}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l})$. If $k' > k$, the new point $(\epsilon_{k'}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l})$ is not necessary because $\frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l}/\epsilon_{k'} > sp^k[j]/\epsilon_k$ which violates the subadditivity constraint. If $k' < k$, the new point $(\epsilon_{k'}, \frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l})$ is also not necessary because $\frac{sp^k[j]}{\epsilon_k}\epsilon_{k_l} > sp^k[j]$ which violates the monotonicity constraint.

Given a survey price point $(\epsilon_k, sp^k[j])$ in model M_k, it determines a $\mathcal{MC}$ point in model M_{k_s}, $(\epsilon_{k_s}, sp^k[j])$ for $k_s < k$. We do not need to generate $\mathcal{MC}$ points based on $(\epsilon_{k_s}, sp^k[j])$ because those $\mathcal{MC}$ points are already determined by $(\epsilon_k, sp^k[j])$. It is also not necessary to generate $\mathcal{SC}$ points based on $(\epsilon_{k_s}, sp^k[j])$ because the $\mathcal{SC}$ points

Algorithm 4: Dynamic programming algorithm for finding an optimal solution of the relaxed revenue maximization problem.

```
input : Model with DP parameter $\epsilon_k$ and its corresponding
        price points in the complete price space.
output: $MAX(\mathcal{RRM})$.
1  for each model $M_k$ do
2      sort the price points in the complete price space in
         decreasing order;
3      use $(\epsilon_k, p^k[j])$ to denote the $j^{th}$ lowest price point;
4      $MR[k, |p^k|] = p^k[|p^k|]f(\epsilon_k, p^k[|p^k|])$;
5      for $j = |p^k| - 1$ to 1 do
6          $MR[k, j] = p^k[j]\sum_{k'=j}^{|p^k|} f(\epsilon_k, p^k[k'])$;
7  for $j = 1$ to $|p^1|$ do
8      $MAX[1, j] = MR[1, p^1[j]]$;
9  for each model $M_k, k = 2, ..., l$ do
10     for each price point $(\epsilon_k, p^k[j])$ do
11         $MAX[k, j] = \max\{MAX[k-1, j']\} + MR[k, j]$,
             where $p^{k-1}[j'] \le p^k[j] \&\& p^{k-1}[j']/\epsilon_{k-1} \ge p^k[j]/\epsilon_k$;
12         $p(L.MAX[k, j]) = p^{k-1}[j']$ that makes $MAX[k, j]$
             achievable in Line 11;
13 $MAX(RRM) = \max\{MAX[l, j]\}$ for $j = 1, ..., |p^l|$;
```

generated by $(\epsilon_{k_s}, sp^k[j])$ are already constrained by the $\mathcal{SC}$ points generated by $(\epsilon_k, sp^k[j])$. □

A recursive solution. We define the revenue of an optimal solution recursively in terms of the optimal solutions to subproblems. We pick as our subproblems the problems of determining the maximum revenue $MAX[k, j]$, where $MAX[k, j]$ denotes the revenue of the optimal solution by considering the first k models and taking the j^{th} lowest price point in the complete price space of model M_k. For the full problem, the maximum revenue would be $\max\{MAX[l, j]\}$ for all the price points $(\epsilon_l, p^l[j])$ in the complete price space in model M_l. For the price points in the complete price space of model M_1, we can directly compute $MAX[1, j]$ for all the price points because there is no initial constraint. For the price points in the complete price space of other models, we need to consider both the monotonicity constraint and the subadditivity constraint. We have a recursive equation as follows.

$$MAX[k, j] = \max\{MAX[k-1, j']\} + MR[k, j], \tag{13}$$

where $p^{k-1}[j'] \le p^k[j] \&\& p^{k-1}[j']/\epsilon_{k-1} \ge p^k[j]/\epsilon_k$ and $MR[k, j]$ denotes the revenue from model M_k if we price model M_k as $p^k[j]$.

Computing the maximum revenue. Now we could easily write a recursive algorithm in Algorithm 4 based on recurrence Eq. 13, where $|p^k|$ is the number of complete price points in model M_k. We compute $MR[k, j]$ for all the price points of the complete price space in Lines 1-6 and $MAX[k, j]$ for the price points in the first model and the other models in Line 8 and Line 11, respectively.

Theorem 3. *Algorithm 4 can be finished in $O(N^2 l^2)$ time.*

Proof. For the $O(N)$ survey price points in the survey price space, we general $O(Nl)$ price points in the complete price space. For each price point in the complete price space, we need $O(Nl)$ time to update $MAX[k, j]$. Therefore, Algorithm 4 requires $O(N^2 l^2)$ time. □

Example 2. *In model M_1 of Figure 2, we have $MAX[1, j]$ for $j = 1, ..., 6$ shown in Table 2. For computing $MAX[2, 2]$, there is only one price point $(1, 3)$ satisfying both the monotonicity constraint and the subadditivity constraint within model M_1. Therefore, we have $MAX[2, 2] = MAX[1, 2] + MR[2, 2] = 3 + 6 = 9$. Similarly, we can fill the entire table shown in Table 2.*

Finding optimal pricing. Although Algorithm 4 determines the maximum revenue of $\mathcal{RRM}$, it does not directly show the optimal price for each model $p(\epsilon_k)$. However, for each price point $(\epsilon_k, p^k[j])$ in the complete price space, we record the price point $p(L.MAX[k, j])$ in model M_{k-1}, which has the maximum revenue in those price points that satisfy both the monotonicity constraint and the subadditivity constraint with $(\epsilon_k, p^k[j])$ in Line 12 of Algorithm 4. Therefore, we can recursively backtrack the optimal price point in model M_{k-1} from the optimal price point in model M_k. We need $O(nl)$ time to find the maximum value in $MAX[l, j]$ and $O(l)$ time to backtrack. In total, we can construct an optimal solution in $O(Nl)$ time. We note that such a solution may be one of several solutions that can achieve the maximum value.

We show a running example in Table 2. We first obtain $MAX[3, 4]$ which has the maximum value among $MAX[3, j]$ for $j = 1, ..., 6$. Therefore, we set $p(\epsilon_3) = 8$. We backtrack to $MAX[2, 4]$ in model M_2 and set $p(\epsilon_2) = 7$. And then we backtrack to $MAX[1, 3]$ in model M_1 and set $p(\epsilon_1) = 4$. Finally, an optimal pricing setting is $\langle p(\epsilon_1), p(\epsilon_2), p(\epsilon_3)\rangle = \langle 4, 7, 8\rangle$.

Table 2: Example for finding optimal pricing.

the j^{th} lowest price point / Model	1	2	3	4	5	6
M_1	2	3	4	0	0	0
M_2	6	9	9	11	4	null
M_3	15	18	19	19	11	4

5.2 Shapley Coverage Maximization Problem for Model Training

Given $D_1, ..., D_n$, the broker builds l versions of model, $M_1, ..., M_l$ under manufacturing budget (maximized revenue in model pricing) and attempts to train the best model for each model version to remain competitive. For each model M_k, each participating data owner D_i receives compensation $c_i(\epsilon)$ (or c_i for the sake of simplicity) when applicable. Under manufacturing budget $\mathcal{MB}$, the broker needs to select a subset $\mathcal{S}$, so that the Shapley coverage of the trained model will be maximized. We formalize the subset selection of $\mathcal{S}$ as a Shapley coverage maximization $\mathcal{SCM}$ problem as follows.

$$\arg\max_{\mathcal{S}\subseteq\{D_1,...,D_n\}} \sum_{i:D_i\in\mathcal{S}} \mathcal{SV}_i, \tag{14}$$

$$s.t. \sum_{i:D_i\in\mathcal{S}} c_i(\epsilon) \le \mathcal{MB}. \tag{15}$$

We prove the above problem is NP-hard in Theorem 4. Given the NP-hard complexity, we present three approximation algorithms. First, we present a pseudo-polynomial time algorithm using dynamic programming. Then, we present a greedy algorithm with the worst case bound if each data owner's compensation is not too large. Finally, we propose an enumerative guess-based greedy algorithm with the worst case bound by relaxing the compensation constraint, which uses the greedy algorithm as a subroutine.

NP-hardness proof. We prove that $\mathcal{SCM}$ is NP-hard by showing that the well-known partition problem is polynomial time reducible to $\mathcal{SCM}$.

Definition 2. ***(Decision Version of $\mathcal{SCM}$)*** *Given a data set $\mathcal{S}$ of n data owners with their corresponding privacy compensation $c_1, ..., c_n$ and Shapley value $\mathcal{SV}_1, ..., \mathcal{SV}_n$, the decision version of $\mathcal{SCM}$ has the task of deciding whether there is a subset $\mathcal{S}_1 \subseteq \mathcal{S}$ such that $\sum_{i:D_i \in \mathcal{S}_1} c_i \leq B$ and $\sum_{i:D_i \in \mathcal{S}_1} \mathcal{SV}_i \geq V$.*

Definition 3. ***(Decision Version of Partition Problem)*** *[23] Given a set $\mathcal{S}$ of n positive integer values $v_1, ..., v_n$, the decision version of partition problem has the task of deciding whether the given set S can be partitioned into two subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ such that the sum of the integers in $\mathcal{S}_1$ equals the sum of the integers in $\mathcal{S}_2$.*

Theorem 4. *The decision version of $\mathcal{SCM}$ is an NP-hard problem.*

Proof. We show that there exists a polynomial reduction by proving that there exists a subset $\mathcal{S}_1 \subseteq \mathcal{S}$ such that $\sum_{i:D_i \in \mathcal{S}_1} c_i \leq B$ and $\sum_{i:D_i \in \mathcal{S}_1} \mathcal{SV}_i \geq V$ if and only if there is a partition $\mathcal{S}_1$ and $\mathcal{S}_2$ such that the sum of the integers in $\mathcal{S}_1$ equals the sum of the integers in $\mathcal{S}_2$. We construct the polynomial reduction as follows. Consider the following instance of $\mathcal{SCM}$: $c_i = v_i$ and $\mathcal{SV}_i = v_i$ for $i = 1, ..., n$, and $B = V = \frac{1}{2} \sum_{i=1}^{n} v_i$.

We show the reduction as follows.

(1) If there exists a partition $\mathcal{S}_1$ and $\mathcal{S}_2$ such that the sum of the integers in $\mathcal{S}_1$ equals to the sum of the integers in $\mathcal{S}_2$, there exists $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\sum_{i:D_i \in \mathcal{S}_1} v_i = \sum_{i:D_i \in \mathcal{S}_2} v_i = \frac{1}{2} \sum_{i=1}^{n} v_i$. We choose the set of data owners $\mathcal{S}_1$ in $\mathcal{SCM}$ and we have $\sum_{i:D_i \in \mathcal{S}_1} c_i = \sum_{i:D_i \in \mathcal{S}_1} v_i = \frac{1}{2} \sum_{i=1}^{n} v_i = B$ and $\sum_{i:D_i \in \mathcal{S}_1} \mathcal{SV}_i = \sum_{i:D_i \in \mathcal{S}_1} v_i = \frac{1}{2} \sum_{i=1}^{n} v_i = V$. Therefore, we know that there exists a subset $\mathcal{S}_1 \subseteq \mathcal{S}$ such that $\sum_{i:D_i \in \mathcal{S}_1} c_i \leq B$ and $\sum_{i:D_i \in \mathcal{S}_1} \mathcal{SV}_i \geq V$.

(2) If there exists a subset $\mathcal{S}_1 \subseteq \mathcal{S}$ such that $\sum_{i:D_i \in \mathcal{S}_1} c_i \leq B$ and $\sum_{i:D_i \in \mathcal{S}_1} \mathcal{SV}_i \geq V$, we partition the set $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1$. We have $\sum_{i:D_i \in \mathcal{S}_1} c_i = \sum_{i:D_i \in \mathcal{S}_1} v_i \leq B = \frac{1}{2} \sum_{i=1}^{n} v_i$ and $\sum_{i:D_i \in \mathcal{S}_1} \mathcal{SV}_i = \sum_{i:D_i \in \mathcal{S}_1} v_i \geq V = \frac{1}{2} \sum_{i=1}^{n} v_i$. This implies that $\sum_{i:D_i \in \mathcal{S}_1} v_i = \frac{1}{2} \sum_{i=1}^{n} v_i$. We also have $\sum_{i:D_i \in \mathcal{S}_2} v_i = \sum_{i=1}^{n} v_i - \frac{1}{2} \sum_{i=1}^{n} v_i = \frac{1}{2} \sum_{i=1}^{n} v_i$. Therefore, there exists a partition $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\sum_{i:D_i \in \mathcal{S}_1} v_i = \sum_{i:D_i \in \mathcal{S}_2} v_i = \frac{1}{2} \sum_{i=1}^{n} v_i$. □

Pseudo-polynomial time algorithm. We first present a pseudo-polynomial time algorithm for $\mathcal{SCM}$. Pseudo-polynomial means that the algorithm has the polynomial time complexity in terms of $\mathcal{MB}$ rather than the traditional number of data owners n. We divide $\mathcal{MB}$ into $\lceil \frac{\mathcal{MB}}{a} \rceil$ parts, where a is the greatest common divisor in c_i for all $i = 1, ..., n$. We define $\mathcal{SV}[i, j]$ as the maximum $\mathcal{SCM}$ that can be attained with manufacturing budget $\leq j \times a$ by only using the first i data owners. The idea is to simplify the complicated problem of computing $\mathcal{SV}[n, \lceil \frac{\mathcal{MB}}{a} \rceil]$ by breaking it down into simpler sub-problems in a recursive manner. The detailed algorithm is shown in Algorithm 5. In Line 5, if the manufacturing budget is not enough, we do not need to consider the i^{th} data owner. Otherwise, we can take D_i if we can get more Shapley value by replacing some data owners from $D_1, ..., D_{i-1}$ in Line 8.

Greedy algorithm. The time cost of Algorithm 5 is extremely dominated by $\mathcal{MB}$ and a. We propose a greedy algorithm based on the ratio between Shapley value $\mathcal{SV}_i$ and compensation c_i in Algorithm 6, which is not sensitive to $\mathcal{MB}$ and a. The idea is to choose data owners with a high $\mathcal{SV}$ but low compensation demand. We sort the data owners in decreasing order of Shapley value per compensation $\mathcal{SV}_i/c_i$ in Line 3. In Lines 6-8, we proceed to take the data owners, starting with as high as possible of $\mathcal{SV}_i/c_i$ until there is no budget. We present a lower bound for Algorithm 6 in Theorem 5, where MAX is the maximum value that we can obtain in Eq. 14.

Algorithm 5: Pseudo-polynomial time algorithm for $\mathcal{SCM}$.

```
input : $c_i$, $\mathcal{MB}$, and $\mathcal{SV}_i$ for $i = 1, ..., n$.
output: $\mathcal{S}$.
1 for $j$=0:$a$:$\mathcal{MB}$ do
2     $\mathcal{SV}[0, j] = 0$;
3 for $i$ =1 to $n$ do
4     for $j$=0:$a$:$\mathcal{MB}$ do
5         if $c_i > j \times a$ then
6             $\mathcal{SV}[i, j] = \mathcal{SV}[i-1, j]$;
7         else
8             $\mathcal{SV}[i, j] =$
                  $max\{\mathcal{SV}[i-1, j], \mathcal{SV}[i-1, j \times a - c_i] + \mathcal{SV}_i\}$;
9 backtrack from $\mathcal{SV}[n, \lceil \frac{\mathcal{MB}}{a} \rceil]$ to $\mathcal{SV}[1, 0]$ to find the
    selected $D_i$;
```

Algorithm 6: Greedy algorithm for $\mathcal{SCM}$.

```
input : $c_i$, $\mathcal{MB}$, and $\mathcal{SV}_i$ for $i = 1, ..., n$.
output: $\mathcal{S}$.
1 for $i$=1 to $n$ do
2     compute $\mathcal{SV}_i/c_i$;
3 sort $\mathcal{SV}_i/c_i$ for $i = 1, ..., n$ in decreasing order and denote as
    $\mathcal{SV}_1/c_1 \geq \mathcal{SV}_2/c_2 \geq ... \geq \mathcal{SV}_n/c_n$;
4 B=0;
5 i=1;
6 while $B \leq \mathcal{MB}$ do
7     add $c_i$ to B;
8     i=i+1;
9 return the corresponding $D_i$ of those $c_i$ in B;
```

Theorem 5. *If for all i, $c_i \leq \zeta\mathcal{MB}$, Algorithm 6 has a lower bound guarantee $(1 - \zeta)MAX$ and can be finished in $O(n \log n)$ time.*

Proof. We set c_k as the first data that is not accepted in Algorithm 6, i.e., we choose the corresponding data owners of $c_1, ..., c_{k-1}$. For $1 \leq i \leq k$, we have $\mathcal{SV}_i/c_i \geq \mathcal{SV}_k/c_k$.

$\Rightarrow \mathcal{SV}_i \geq c_i \frac{\mathcal{SV}_k}{c_k}$

$\Rightarrow \mathcal{SV}_1 + ... + \mathcal{SV}_k \geq (c_1 + ... + c_k) \frac{\mathcal{SV}_k}{c_k}$

Because we set c_k as the first data that is not accepted, i.e., $c_1 + ... + c_k > \mathcal{MB}$, we have

$\Rightarrow \mathcal{SV}_k \leq (\mathcal{SV}_1 + ... + \mathcal{SV}_k) \frac{c_k}{\mathcal{MB}}$

$\Rightarrow \mathcal{SV}_k \leq \zeta(\mathcal{SV}_1 + ... + \mathcal{SV}_k)$

$\Rightarrow \mathcal{SV}_k \leq \frac{\zeta(\mathcal{SV}_1 + ... + \mathcal{SV}_{k-1})}{1-\zeta}$

Because $\mathcal{SV}_1 + ... + \mathcal{SV}_k \geq MAX$, we have $\mathcal{SV}_1 + ... + \mathcal{SV}_{k-1} \geq (1 - \zeta)MAX$. Therefore, Algorithm 6 has a lower bound guarantee $(1 - \zeta)MAX$.

The time complexity is dominated by the sorting in Line 3. Therefore, Algorithm 6 can be finished in $O(n \log n)$ time. □

Lemma 3. *There are at most $\lceil \frac{1}{\alpha} \rceil$ data owners having compensation c_i such that their corresponding Shapley value $\mathcal{SV}_i$ is at least αMAX in any optimal solution.*

Lemma 3 is easy to see, otherwise, the optimal solution value is larger than MAX, which is a contradiction.

Enumerative guess-based greedy algorithm. Although Algorithm 6 can achieve $(1 - \zeta)MAX$, the requirement of $c_i \leq \zeta\mathcal{MB}$ is a

Algorithm 7: Enumerative guess-based greedy algorithm for $\mathcal{SCM}$.

```
input : $c_i$, $\mathcal{MB}$, and $\mathcal{SV}_i$ for $i = 1, ..., n$.
output: $\mathcal{S}$.
1  for i=1 to h do
2      choose $i$ data owner(s) to compose a subset $\mathcal{S}'$;
3  we have $\sum_{i=1}^{h} \binom{n}{i}$ such subsets;
4  for $j$=1 to $\sum_{i=1}^{h} \binom{n}{i}$ do
5      compute the manufacturing budget of the data owners in $\mathcal{S}'$;
6      delete those $\mathcal{S}'$ if their manufacturing budget is larger than $\mathcal{MB}$;
7  we have $r$ remaining subsets $\mathcal{S}'_1, \mathcal{S}'_2, ..., \mathcal{S}'_r$;
8  for each subset $\mathcal{S}'_j$, $j = 1, ..., r$ do
9      let $D_a$ be the data owner with the least Shapley value in $\mathcal{S}'_j$, remove all data owners in $\mathcal{S}_j - \mathcal{S}'_j$ if their Shapley value is larger than $\mathcal{SV}_a$ and get a new subset $\mathcal{S}''_j$;
10     run Algorithm 6 in $\mathcal{S}''_j$ with remaining manufacturing budget $\mathcal{MB} - \sum_{i=1}^{|\mathcal{S}'_j|} c_i$;
11 return the data owners in $\mathcal{S}'_j$ and $\mathcal{S}''_j$, where $\mathcal{S}'_j$ and $\mathcal{S}''_j$ have the highest Shapley value among $j = 1, ..., r$;
```

little strict. We present another algorithm with the same worst case bound but without the above requirement. The idea is to guess the h most profitable data owners in an optimal solution and compute the rest greedily as in Algorithm 6. The detailed algorithm is shown in Algorithm 7. Let $\alpha \in (0, 1)$ be a fixed constant and $h = \lceil \frac{1}{\alpha} \rceil$. We first enumerate all the subsets with data owner size $\leq h$ in Lines 1-3. We delete those subsets with higher manufacturing budget than $\mathcal{MB}$ in Lines 4-6. In Lines 7-10, for each remaining subset, we call Algorithm 6 to maximize the Shapley value with the remaining budget after taking the $\leq h$ data owners.

Theorem 6. *Algorithm 7 runs in $O(n^{\lceil \frac{1}{\alpha} \rceil})$ time with $(1-\alpha)$MAX worst case bound.*

Proof. For the time complexity, we have at most $\sum_{i=1}^{h} \binom{n}{i}$ subsets $\mathcal{S}'$ after deleting those subsets if their compensation sum is larger than $\mathcal{MB}$. That is, we have at most n^h different subsets $\mathcal{S}'$. For each subset $\mathcal{S}'$, the greedy Algorithm 6 only requires linear time to handle the remaining data owners. Therefore, the total time cost for Algorithm 7 is $O(n^{\lceil \frac{1}{\alpha} \rceil + 1})$.

For the worst case approximation bound, we assume subset $\mathcal{S}'$ in the optimal solution has exact h data owners. We note that subset $\mathcal{S}'$ in the optimal solution may have $\leq h$ data owners, but it is easy to see that this does not affect the complexity analysis. If the number of data owners in the optimal solution is less than h, the optimal solution will be included in $\mathcal{S}'$. In the following, we discuss the case that the number of data owners in the optimal solution is larger than h.

We have $h + k$ data owners $D_1, ..., D_h, D_{h+1}, ..., D_{h+k-1}, D_{h+k}$ that need to be considered, where $D_1, ..., D_h$ are the data owners in subset $\mathcal{S}'$, D_{h+i} is the i^{th} data owner with the highest $\mathcal{SV}_i/c_i$ in $\mathcal{S}''$. D_{h+k} is the data owner with the highest $\mathcal{SV}_i/c_i$ rejected by the greedy algorithm of Algorithm 6. Let MAX' be the maximum value for the data owners in $\mathcal{S}''$. Therefore, we have $\mathcal{SV}(\mathcal{S}'') + \mathcal{SV}_{h+k} \geq MAX'$.

$\Rightarrow\ \mathcal{SV}(\mathcal{S}'') \geq MAX' - \mathcal{SV}_{h+k}$

Based on Lemma 3, there are at most $\lceil \frac{1}{\alpha} \rceil$ data owners having compensation c_i such that their corresponding Shapley value $\mathcal{SV}_i$ is at least αMAX in any optimal solution, and those $\lceil \frac{1}{\alpha} \rceil$ data owners are already pruned in Line 9. Therefore, we have $\mathcal{SV}_{h+k} \leq \alpha MAX$.

$\Rightarrow\ \mathcal{SV}(\mathcal{S}'') \geq MAX' - \alpha MAX$

$\Rightarrow\ \mathcal{SV}(\mathcal{S}') + \mathcal{SV}(\mathcal{S}'') \geq MAX' + \mathcal{SV}(\mathcal{S}') - \alpha MAX$

$\Rightarrow\ \mathcal{SV}(\mathcal{S}') + \mathcal{SV}(\mathcal{S}'') \geq MAX - \alpha MAX$

That is, Algorithm 7 has the worst case bound $(1-\alpha)MAX$. □

Algorithm 8: The complete *Dealer* dynamics.

```
1  %% Data collection;
2  collect dataset $\{D_1, ..., D_n\}$ along with compensation function $c_i(\epsilon)$ for $i = 1, ..., n$ based on Shapley value computed in Algorithm 1;
3  %% Model parameter setting;
4  decide a set of $l$ models to train with coverage rate $\mu_k$ and DP parameter $\epsilon_k$ for $k = 1, ..., l$;
5  %% Model pricing;
6  perform market survey among $m'$ sampled model buyers (survey participants) based on each model buyer's pricing function $P(B_j, M_k)$: collect market survey results of model demand $tm_j$ and valuation $v_j$ for $j = 1, ..., m'$;
7  call Algorithms 3 and 4 to compute the optimal price $p(\epsilon_k)$ of model $M_k$ for $k = 1, ..., l$;
8  %% Model training and release;
9  for k=1 to l do
10     data selection: call Algorithm 5, 6, or 7 to select training subset $\mathcal{S}^k$ with manufacturing budget $p(\epsilon_k) \cdot purchase(p(\epsilon_k))$ to maximize $\mathcal{SV}(\mathcal{S}^k)$;
11 if for all k, $\frac{\mathcal{SV}(\mathcal{S}^k)}{\mathcal{SV}(\{D_1,...,D_n\})} \geq \mu_k$ then
12     model training: train the model with subset $\mathcal{S}^k$ by Algorithm 2;
13     model release: release model $M_k$, its coverage ratio $\mu_k$, DP parameter $\epsilon_k$, and model price $p_k$;
14 %% Model transaction;
15 model buyers send their target models and payment to the broker, and the broker sends the corresponding models to model buyers.
16 %% Compensation allocation;
17 for k=1 to l do
18     allocate the corresponding compensation to $D_i$ based on its compensation function $c_i(\epsilon_k)$ if $D_i$ is used to train Model $M_k$;
```

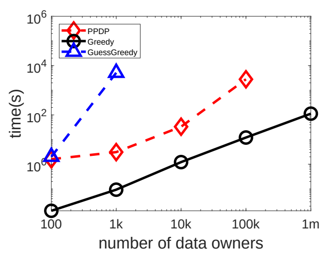
5.3 Complete Dealer Dynamics

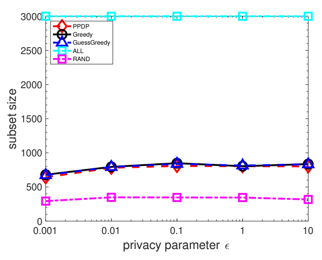
We summarize the complete *Dealer* dynamics in Algorithm 8, which integrates all the proposed algorithms in the previous sections. We assume that the broker can set appropriate parameters l, ϵ_k, and μ_k based on her market experiences. If the maximized revenue cannot cover the stated Shapley coverage rates, the broker needs to reset the parameters for the l models.

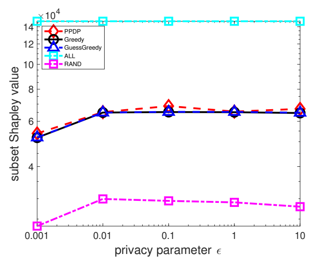
6. EXPERIMENTS

In this section, we present experimental studies validating: 1) our proposed algorithms for pricing models can generate more revenue for the data owners and the broker, and significantly outperform the baseline algorithms in terms of efficiency, and 2) our proposed algorithms for subset selection in model training are efficient and effective;.

6.1 Experiment Setup


(a) time cost


(b) subset size


(c) Shapley value

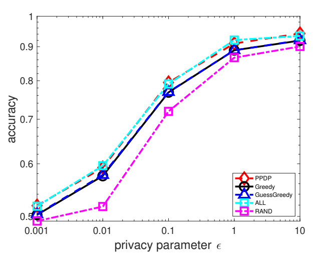

(d) accuracy

Figure 3: Model training.

We ran experiments on a machine with an Intel Core i7-8700K and two NVIDIA GeForce GTX 1080 Ti running Ubuntu with 64GB memory. We employed Support Vector Machine (SVM) as our machine learning model and used both synthetic datasets and a real chess dataset [14] in our experiments. The chess dataset includes 3196 data tuples, each having 36 attributes.

For model pricing, we compare our proposed algorithm with several baseline algorithms as follows.

- **DPP**: applying the dynamic programming algorithm in Algorithm 4 to the survey price space.
- **DPP+**: applying the dynamic programming algorithm in Algorithm 4 to the complete price space.
- **Linear**: taking the lowest survey price from model M_1 and the highest survey price from model M_M and using linear interpolation for the remaining models $M_2, ..., M_{M-1}$ based on the two end-prices.
- **Low**: taking the lowest price in each model.
- **Median**: taking the median price in each model.
- **Base**: applying the exhaustion-based approach to the survey price space.
- **Base+**: applying the exhaustion-based approach to the complete price space.

For model training, we compare the three subset selection algorithms we proposed with two baseline algorithms as follows.

- **PPDP**: the pseudo-polynomial dynamic programming algorithm in Algorithm 5.
- **Greedy**: the greedy algorithm in Algorithm 6.
- **GuessGreedy**: the guess based greedy algorithm in Algorithm 7.
- **ALL**: selecting the entire chess training dataset.
- **RAND**: randomly selecting a subset of the entire tuples given the manufacturing budget.

6.2 Model Pricing

Efficiency. We experimentally study the efficiency of the proposed algorithms for pricing models. Because Linear, Low, and Median algorithms only need to scan the survey price points once, the time cost is very low. For the ease of presentation, we omit the experimental results for the three algorithms.

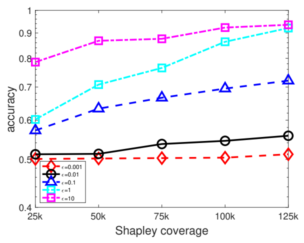


Figure 4: Accuracy.

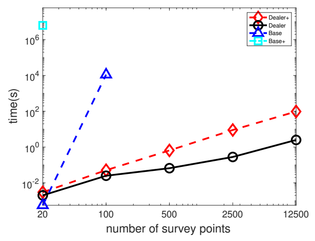


Figure 5: Time cost.

Figure 5 shows the time cost of DPP, DPP+, Base, and Base+ on varying number of survey price points, i.e., the number of surveyed model buyers. Both DPP and DPP+ linearly increase with the number of survey price points, which verifies the efficiency of the proposed dynamic programming algorithm. The time cost of both Base and Base+ is prohibitively high due to the enormous volume of the price combinations for different models.

Effectiveness. We experimentally study the revenue gain of the proposed algorithms on datasets with different distributions. We omit Base+ due to the prohibitively high cost. We generate two datasets with 100 survey price points according to simulated price functions and Eq. 8 (i.e., from 100 model buyers). The number of survey price points on each model follows uniform distribution and Gaussian (mean=3, standard deviation=1.2) distribution, respectively. For the first model of both datasets, we generate the survey price points following uniform distribution with range $[1000, 5000]$. The remaining four models follow a 1000 increase on both the lower bound and the upper bound of the range.

Figures 6(a)(b)(c)(d) show the survey price points, pricing setting, affordability ratio (fraction of the model buyers that can afford to buy a model), and revenue on a uniformly distributed dataset, respectively. Figure 6(a) shows the survey price points on 5 models with differential privacy parameters $\epsilon = 0.001, 0.01, 0.1, 1,$ and 10, respectively. Figure 6(b) shows that DPP+ and DPP have similar optimal price settings. All models in DPP have different prices. For the price setting of DPP+, the second model and the third model have the same price, which maximizes the revenue compared to DPP and verifies the effectiveness of the complete price space construction. Figure 6(c) shows that DPP+ has the highest affordability ratio except for Low. For the most critical metric revenue, DPP+ outperforms the other algorithms at least 3%, which verifies the gain of the complete price space construction. Furthermore, DPP has the same performance with Base, which has the optimal revenue. That is, although we relaxed the constraint of $p(\epsilon_k) + p(\epsilon_{k'}) \geq p(\epsilon_k + \epsilon_{k'})$ in Eq.10 to $p(\epsilon_k)/\epsilon_k \geq p(\epsilon_{k'})/\epsilon_{k'}$, the relaxed problem has the same optimal revenue with the original problem.

In practical applications, it is more likely that the survey price points follow Gaussian distribution rather than uniform distribution following the distribution of the noise expectation η_j in the price functions of model buyers. Figures 7(a)(b)(c)(d) show that all algorithms have similar performances on Gaussian distribution dataset as on uniform distribution dataset.

6.3 Model Training

Efficiency. Figures 3(a) shows the time cost of Greedy, PPDP, and GuessGreedy with varying number of data owners on synthetic datasets. PPDP has the intermediate time cost among the three algorithms. Greedy significantly outperforms both PPDP and GuessGreedy due to its simplicity. GuessGreedy costs the highest time because it needs to enumerate $\binom{n}{h}$ subsets, where n is the total number of data owners and h is the size of the sampled subsets during enumeration. In our experiments, the time cost for GuessGreedy is prohibitively high even when we set $h = 2$. We skip some results of PPDP and GuessGreedy in the figure due to their high cost.

Effectiveness. We experimentally study the effectiveness of the proposed subset selection algorithms PPDP, Greedy, and GuessGreedy. We simulate five models by adding DP noise with parameters $\epsilon = 0.001, 0.01, 0.1, 1,$ and 10 in the training processing following Algorithm 2, and each model has $\mathcal{MB}^k = \epsilon_k/\epsilon_1 \sum_{i=1}^{n} c_i(\epsilon_k)$ for $k = 1, ..., 5$, respectively.

Figure 3(b) shows the number of tuples selected by PPDP, Greedy, GuessGreedy, ALL, and RAND, respectively. We employed 3000 tuples as the training dataset. Given the manufacturing budget, our proposed algorithms select more tuples than RAND. Figure 3(c) shows the sum of Shapley value of the tuples in the subsets selected by PPDP, Greedy, GuessGreedy, ALL, and RAND, respectively. We used the absolute value of the Shapley value in order to select those data points that affect the model performance both positively and negatively. Because ALL has the entire tuples, ALL has the highest sum of Shapley value. All the sum of Shapley value of the tuples selected by our proposed algorithms, PPDP, Greedy, and GuessGreedy, is higher than RAND. Furthermore, PPDP outperforms Greedy and GuessGreedy, which verifies the effectiveness of the proposed dynamic programming algorithm. Figure 3(d) shows the accuracy of the models trained on the subsets selected by PPDP, Greedy, GuessGreedy, ALL, and RAND, respectively. Given the manufacturing budget, our proposed subset selection algorithms only choose around 800 tuples. However, the accuracy of the proposed algorithms is even higher than ALL, which verifies the effectiveness of Shapley value in selecting important data tuples for training high utility model. Comparing the proposed algorithms, PPDP has the highest accuracy with the corresponding highest sum of Shapley value in Figure 3(c). That is, given the limited manufacturing budget, we can choose the subset with the higher sum of Shapley value to obtain the model with higher accuracy. We also can see that the accuracy is predominantly determined by DP parameter ϵ. Figure 4 shows the relationship between Shapley coverage and accuracy. We can see that the accuracy increases with respect to the increasing Shapley coverage in general.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed the first end-to-end model marketplace with differential privacy *Dealer*. Data owners specify their desired compensation functions based on privacy sensitivity and Shapley value. If the compensation proposed by data owner D_i is too high, there is a risk that the broker won't choose D_i to train the model. Model buyers provide their price functions that they are willing to pay based on Shapley coverage sensitivity and noise sensitivity. If the price provided by model buyer B_j is too low, there is a risk that B_j cannot purchase the model from the broker. Based on the compensation functions from data owners and the price functions from model buyers, the broker builds l versions of models with different Shapley coverage rates and DP parameters, and presents an arbitrage-free model pricing mechanism to price the model versions to maximize the revenue (model pricing). Given the maximized revenue, the broker maximizes the Shapley coverage for each model version to remain competitive (model training). We proposed efficient algorithms for both optimization problems of model pricing and model training. Experimental results on the real chess dataset and synthetic datasets show that the proposed algorithms for model pricing and model training are efficient and effective.

The final accuracy depends on the model buyers' test datasets. That is also one of the reasons that we used data coverage instead of model accuracy as the utility metric, since it measures the relative utility of using a subset instead of the entire data from all data owners for training the model. There are several ways to address the practical issues as mentioned. The market survey can include other options for the model buyer to select the model that interests her, such as decision trees. The target model can be sent to the model buyer with a user manual on hyper parameter tuning by the broker.

8. REFERENCES

[1] Dawex, https://www.dawex.com/en/.

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

[3] https://support.gnip.com/apis/.

[4] https://www.bloomberg.com/professional/product/market-data/.

[5] Iota, https://data.iota.org/.

[6] Safegraph, https://www.safegraph.com/.

[7] A. Agarwal, M. Dahleh, and T. Sarkar. A marketplace for data: An algorithmic solution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 701–726. ACM, 2019.

[8] M. Ancona, C. Öztireli, and M. H. Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *ICML*, pages 272–281, 2019.

[9] J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & OR*, 36(5):1726–1730, 2009.

[10] G. C. Cawley and N. L. C. Talbot. Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. *Pattern Recognition*, 36(11):2585–2592, 2003.

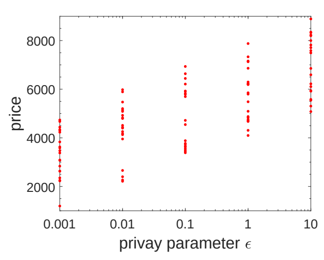
[11] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[12] S. Chawla, S. Deep, P. Koutris, and Y. Teng. Revenue maximization for query pricing. *PVLDB*, 13(1):1–14, 2019.

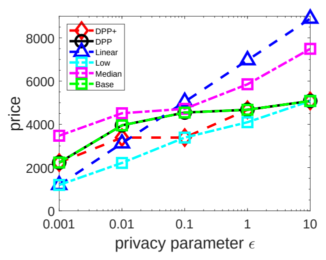
[13] L. Chen, P. Koutris, and A. Kumar. Towards model-based pricing for machine learning in a data marketplace. In *SIGMOD*, pages 1535–1552, 2019.

[14] D. Dua and C. Graff. UCI machine learning repository, 2017.

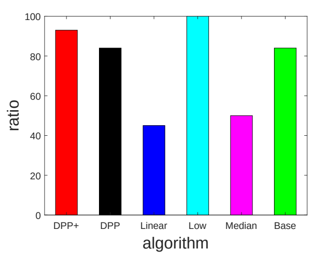
[15] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.



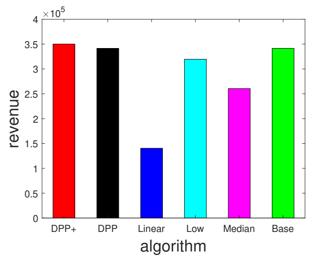
(a) survey data



(b) model price

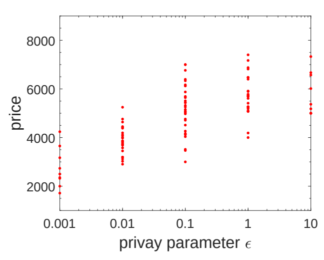


(c) ratio

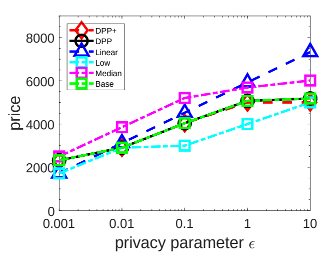


(d) revenue

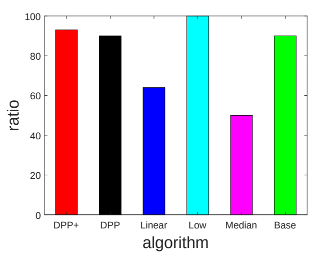
Figure 6: Independent distribution.



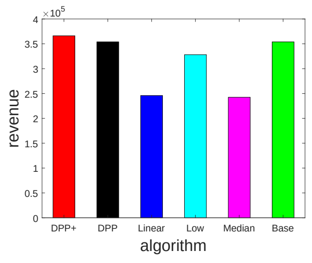
(a) survey data



(b) model price



(c) ratio



(d) revenue

Figure 7: Gaussian distribution.

[16] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[17] S. S. Fatima, M. J. Wooldridge, and N. R. Jennings. A linear approximation method for the shapley value. *Artif. Intell.*, 172(14):1673–1699, 2008.

[18] A. Ghorbani and J. Y. Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, pages 2242–2251, 2019.

[19] A. Ghosh and A. Roth. Selling privacy at auction. In *Proceedings 12th ACM Conference on Electronic Commerce (EC-2011), San Jose, CA, USA, June 5-9, 2011*, pages 199–208, 2011.

[20] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *SODA*, pages 1164–1173, 2005.

[21] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang. Towards practical differentially private convex optimization. In *IEEE S and P*.

[22] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. M. Gurel, B. Li, C. Zhang, C. Spanos, and D. Song. Efficient task-specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment*, 12(11):1610–1623, 2019.

[23] R. E. Korf. A complete anytime algorithm for number partitioning. *Artif. Intell.*, 106(2):181–203, 1998.

[24] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Query-based data pricing. In *PODS*, pages 167–178. ACM, 2012.

[25] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Toward practical query pricing with querymarket. In *SIGMOD*, pages 613–624. ACM, 2013.

[26] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Query-based data pricing. *J. ACM*, 62(5):43:1–43:44, 2015.

[27] C. Li, D. Y. Li, G. Miklau, and D. Suciu. A theory of pricing private data. In *ICDT*, pages 33–44, 2013.

[28] C. Li, D. Y. Li, G. Miklau, and D. Suciu. A theory of pricing private data. *ACM Trans. Database Syst.*, 39(4):34:1–34:28, 2014.

[29] C. Li, D. Y. Li, G. Miklau, and D. Suciu. A theory of pricing private data. *Commun. ACM*, 60(12):79–86, 2017.

[30] B. Lin and D. Kifer. On arbitrage-free pricing for general data queries. *PVLDB*, 7(9):757–768, 2014.

[31] C. Shapiro and H. Varian. Versioning: The smart way to sell information. *Harvard Business Review*, 76(6):107–115, 1998.

[32] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.