Mining Revenue-Maximizing Bundling Configuration

Loc Do
School of Information Systems
Singapore Management University
hldo.2012@smu.edu.sg

Hady W. Lauw
School of Information Systems
Singapore Management University
hadywlauw@smu.edu.sg

Ke Wang*
School of Computing Science
Simon Fraser University
Canada
wangk@cs.sfu.ca

ABSTRACT

With greater prevalence of social media, there is an increasing amount of user-generated data revealing consumer preferences for various products and services. Businesses seek to harness this wealth of data to improve their marketing strategies. Bundling, or selling two or more items for one price is a highly-practiced marketing strategy. In this paper, we address the bundle configuration problem from the data-driven perspective. Given a set of items in a seller's inventory, we seek to determine which items should belong to which bundle so as to maximize the total revenue, by mining consumer preferences data. We show that this problem is NP-hard when bundles are allowed to contain more than two items. Therefore, we describe an optimal solution for bundle sizes up to two items, and propose two heuristic solutions for bundles of any larger size. We investigate the effectiveness and the efficiency of the proposed algorithms through experimentations on real-life rating-based preferences data.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.8 [**Database Applications**]: Data Mining

Keywords

bundling; revenue maximization; willingness to pay

1. INTRODUCTION

With the increasing prevalence and richness of social media, there is a fast-growing wealth of data about consumer preferences. For instance, a consumer rates an item or writes an online review for various products and services, be it a consumer electronic on Amazon, a hotel on TripAdvisor, or a restaurant on Yelp. There are also rich consumption data that reveal the implicit preferences of consumers, such as the amount of time a user spends listening to various songs, or watching various videos. Businesses are interested in harnessing this wealth of data on consumer preferences to estimate market demand, as well as to learn insights that help them in catering their offerings closer to consumer wants.

One marketing strategy that businesses frequently engage in is *bundling*, or selling two or more items for one price. This practice is pervasive across many industries. Comcast and many other cable television companies worldwide sell subscriptions not only for individual channels, but also for bundles of channels. Telecommunication providers, such as AT&T, frequently offer a bundle of services, including cable, telephone, and Internet subscriptions. Travel packages commonly bundle airfare, hotel stay, and attractions.

One important issue is how to design bundles based on consumer demand, so as to achieve a business objective, e.g., maximizing profit or revenue. Given N items, how would a seller determine which items to sell as a bundle, and at what price? We call this the *bundle configuration problem.*

The traditional business approach is to rely on preferences elicited directly from consumers, e.g., through surveys or market research [25]. This approach does not scale up well due to both the cardinality M of consumers whose preferences are to be elicited, and the cardinality N of items to be bundled. For N items, there could be up to $2^N - 1$ possible bundles (including "bundles" containing a single item). Moreover, to obtain a bundle configuration, we further need to consider various combinations of these possible bundles.

The large scale of data poses significant data management and computational challenges. To elicit consumer preferences in a scalable manner, we propose a data-driven perspective, by mining the ever available preference data, such as ratings and reviews on the Internet. To determine the optimal bundle configuration in a tractable manner, we seek efficient algorithms to find the bundle configuration maximizing a utility function based on consumer preferences.

Utility Maximization. We introduce several terminologies to help define the objective of bundling. Willingness to pay is the maximum amount that a consumer is willing to give up in exchange for an item. We represent consumer u's willingness to pay for an item A as $w_{u,A} \in \mathbb{R}_0^+$. The unit can be any measurement of value, such as dollars. In Section 3.1, we discuss how willingness to pay can be estimated from consumer preferences. Table 1 shows an example of three consumers u_1, u_2, and u_3 and two items A and B. For instance, u_1 is willing to pay \$12 for A, whereas u_2 and u_3 are willing to pay only \$8 and \$5 respectively.

*This work was done partially while the author was visiting Singapore Management University. The work is partially supported by a Discovery Grant of the Natural Sciences and Engineering Research Council, Canada.

Proceedings of the VLDB Endowment, Vol. 8, No. 5

Consumer	Willingness to Pay			Components		Pure Bundling	Mixed Bundling		
	$w_{u,A}$	$w_{u,B}$	$w_{u,AB}$	$p_A = \$8.00$	$p_B = \$11.00$	$p_{AB} = \$15.20$	$p_A = \$8.00$	$p_B = \$11.00$	$p_{AB} = \$15.20$
u_1	\$12.00	\$4.00	\$15.20	✓		✓			✓
u_2	\$8.00	\$2.00	\$9.50	✓			✓		
u_3	\$5.00	\$11.00	\$15.20		✓	✓			✓
Revenue				*\$27.00*		*\$30.40*	*\$38.20*		

Table 1: Positive Example of Bundling

For any one item, the seller charges all consumers the same price. A transaction occurs when the buyer's willingness to pay exceeds or equals the price [1]. For Table 1, the price of A is $p_A = \$8$, which results in u_1 and u_2 purchasing A, earning the seller a *revenue* of $2 \times \$8 = \$16$. Since u_1 values the item higher than the price, u_1 obtains a *consumer surplus* of $\$12 - \$8 = \$4$. It follows that u_2 does not enjoy any consumer surplus. Because u_3's willingness to pay of \$5 is below p_A, she does not engage in a transaction, which benefits neither u_3 nor the seller. This uncaptured revenue of \$5 is *deadweight loss*. This example assumes that each consumer demands up to one unit of each item, and the seller has an unlimited supply (see Section 3.2).

We approach the problem from the seller's point of view. For the seller, both profit and consumer surplus are important and necessary to keep the business going. We adopt the objective of maximizing the following utility function, where $\alpha \in [0, 1]$ is the relative weight between the two factors:

$$\alpha \times profit + (1 - \alpha) \times surplus$$

Without loss of generality, in the following discussion we assume that $\alpha = 1$, i.e., maximizing profits while still leaving some consumers with surplus (from higher willingness to pay than the price). Our technique will still apply for any α.

To maximize profit, we need to maximize revenue minus costs. While there are two types of costs (fixed cost and variable cost), only variable cost is affected by the quantity sold (which in turn is affected by pricing), i.e.,

$$profit \propto \sum_{each\ bundle} quantity\ sold \times (price - variable\ cost)$$

If each consumer requires one unit (see Section 3.2), the quantity sold is the number of consumers purchasing the item. When the variable cost is zero or very small, such as in certain markets like digital goods (e.g., cable TV, video on demand) [3], profit maximization is equivalent to revenue maximization. We will focus on such cases. This is without loss of generality, because the technique described can still use profit maximization when the variable costs are known.

While we frame the discussion of "utility" in monetary terms, the only assumption is that utility is additive. In some scenarios, e.g., data marketplaces, the objective may be non-monetary. An example of data marketplace is where the "seller" is a Data as a Service (DaaS) provider who provides content to consumers. Bundling corresponds to grouping correlated data and content (such as selling a hotel list and a review database), or data sets and related analysis reports, etc. Non-monetary utility may be "user satisfaction", measured by the aggregate preference for a group of items.

Bundling Configuration. Suppose the seller is dealing with an inventory of N items. One avenue towards a higher revenue is through bundling. There are three main bundling strategies [1], which we will illustrate using the example willingness to pay shown in Table 1.

- *Components* is when the seller sells only the individual items. Revenue maximization leads to the optimal prices $p_A = \$8$ and $p_B = \$11$, which result in \$16 of revenue from A (earned from u_1 and u_2) and \$11 of revenue from B (earned from u_3), for a total of \$27.
- *Pure bundling* is when the seller sells only the bundle. u_1 is willing to pay $w_{u_1,AB} = \$15.20$ for the bundle $\{A, B\}$, which is the same as u_3. u_2 is willing to pay \$9.50. The optimal price is therefore $p_{AB} = \$15.20$, which nets a revenue of \$30.40 (from u_1 and u_2).
- *Mixed bundling* is when the seller makes both the bundle and the components available. Consumers choose to buy any component (A or B), or the bundle $\{A, B\}$, or none. The seller thus can capture additional revenue from consumers who is willing to pay only for one item (e.g., u_2), resulting in a total revenue of \$38.20.

In this example, both pure and mixed bundling result in higher revenues than *Components*. One explanation concerns the variance in willingness to pay between different users for the same item. This variance results in a flatter distribution of combined willingness to pay for the bundle. However, bundling does not always increase revenue. Whether it does so requires inspection of the willingness to pay. The large numbers of items and consumers lead to a huge number of potential bundles, and a combinatorial explosion of various combinations of bundles.

Contributions. We make the following contributions.

- We advocate addressing the bundle configuration problem by mining willingness to pay from the ever-growing and available user-generated preference data, such as ratings. This is a major difference from the traditional business approach where explicit solicitation from consumers is frequently required, which does not work in large scale.
- We make a major algorithmic contribution towards addressing the computational complexity of the bundling problem without pre-fixing the search space. In Section 3, we formulate the k-sized bundling problem for any k. We show that optimal pure bundling configuration is NP-hard for bundle sizes greater than 2. Moreover, we propose two efficient heuristic solutions in Section 5 that apply for both pure bundling and mixed bundling.
- To model the uncertainty in adoptions, we design a stochastic model for adoptions in Section 4, which generalizes the deterministic step function used in previous work.
- We comprehensively validate the efficacies of our proposed approach through experiments on user-generated rating data in Section 6.

2. RELATED WORK

Bundling is of interest in both management sciences (e.g., economics and marketing), as well as computer science.

Economics, Marketing, and Management Sciences. The seminal work in [1] established bundling as an effective device in a market with diverse consumers' willingness to pay. This is followed by a body of subsequent works investigating bundling strategies as covered by the more recent survey in [28]. A focus of these works is on the study of factors related to pricing, such as heterogeneity in consumers' reservation prices, extent of correlation of pricing, complementarity or substitutability [27], nature of competition. Other factors include varying prices based on the bundle size [10].

In comparison to these works, our work has two main distinctions. For one, these works rely on traditional solicitation of consumers' willingness to pay. Inspired by the wide availability of customer preference data online, we rely on mining these preferences from sources such as online ratings. For another, most of these works focus only on size-2 bundles, and few consider larger bundles, which are computationally much more challenging. For instance, [19] considers very small problem sizes (five to ten items). In contrast, we solve the utility-maximizing bundling problem without prefixing the search space, and directly address the scalability of considering a much larger number of items.

Computer Science. We also review related work in computer science on bundling and utility maximization.

In terms of bundling. One category of related work is concerned with finding a group of items maximizing a certain objective, or satisfying certain constraints. Product design deals with finding an optimal configuration of features in designing a new product [4]. In [12], the problem is to design a new item (or k new items) that can attract the maximum number of the tags specified in a query. High utility itemset mining [24] returns itemsets with total utility larger than a pre-defined threshold. Yet another set of works consider grouping items based on compatibility [26], for instance constructing composite itemsets including one main product and its accessories [6]. These works are mostly attribute-oriented, while ours is a pricing-oriented problem, where the optimization objective depends not just on how we group items, but also on how we price the bundles.

Bundle recommendation [30, 23] recommends a group of items that maximize a user's aggregate preferences for those items. A key distinction is that bundling aims at maximizing profit by considering willingness to pay and prices, whereas recommendation considers rating data and does not have a mechanism like pricing to model adoptions.

In terms of utility maximization. There are non-bundling ways to increase revenue. In combinatorial auction [13], each bidder places bids for a subset of the items. Since the auctioneer has only one unit for each item, the problem is to find an allocation of items to bidders that maximizes the auctioneer's revenue. Our work considers a different market setting where the seller has multiple units to serve multiple consumers. Another line of work that considers a similar market setting as ours is the pricing problem [5, 17], which is concerned with determining prices for single items, based on various assumptions on which subset of items are desired by each consumer. Unlike ours, such works do not deal with the computational complexity of forming bundles.

3. OVERVIEW

We consider a set of M consumers $U = \{u_1, \ldots, u_M\}$ and a set of N items or components $I = \{i_1, \ldots, i_N\}$. W is an $M \times N$ matrix, where each element $w_{u,i} \in \mathbb{R}_0^+$ indicates how much a consumer u is willing to pay for an item i.

3.1 Willingness to Pay

In this work, we adopt a data-driven approach to bundle configuration by mining willingness to pay from consumer preferences data. Estimating willingness to pay is complex, and encompasses a whole research area [7]. Broadly speaking, the major approaches are *revealed preference* (actions such as auction bids or purchase transactions) and *stated preference* (statements such as surveys and ratings). The former requires transaction data (not widely available). The latter, with methods such as conjoint analysis [16] and the Nobel prize-winning discrete choice theory [25], relies on careful experiments to elicit consumer responses. We adopt the stated preference approach, but instead of relying on direct elicitation of consumer responses (which is not scalable), we rely on the wealth of user-generated data indicative of consumer preferences. In particular, we describe how the matrix W can be derived from ratings data in Section 6.1.1.

Since the matrix W only specifies the willingness to pay for the individual items, we also need to define the willingness to pay for a bundle. Suppose we define a bundle $b \subseteq I$ as a set of items. Ideally, a consumer u's willingness to pay for a bundle b, denoted $w_{u,b}$, can be obtained directly as well. This is however impractical and too demanding, as it requires obtaining willingness to pay for all possible bundles. The current accepted approach is to derive $w_{u,b}$ as a function of the willingness to pay for individual items in the bundle, i.e., $\{w_{u,i}\}_{i \in b}$. In this work, we adopt the function proposed by [27], which is shown in Equation 1.

$$w_{u,b} = \sum_{i \in b} w_{u,i} \times (1 + \theta) \tag{1}$$

The bundling coefficient θ models the interaction between the items being bundled. If the items are *independent* (e.g., a book and a toaster), then the willingness to pay for one item does not affect the others, i.e., $\theta = 0$. This is the setting used in most of the previous works [1]. If the items are *substitutes* (e.g., two fiction books), then $\theta < 0$, and the willingness to pay for the bundle would be less than paying for both items individually. This scenario is probably quite common, as different items (especially non-necessities) are to some extent substitutes for one another. For the example in Table 1, the bundling coefficient is $\theta = -0.05$. If the items are *complementary* (e.g., ski rental and training), then $\theta > 0$, i.e., a consumer has higher willingness to pay for both items together than separately.

The appropriate setting of θ is correlated to the issue of compatibility between different items, which is an orthogonal research issue. Compatibility is a focus of study in bundling literature in economics and marketing, in the form of complementarity or substitutability [27]. With the parameter θ, our framework could leverage the results in that literature. Here, we do not make any assumption about a specific θ, and study various θ settings later in Section 6. It is also advantageous to have a θ that can be specified for a marketing or domain expert to simulate the revenue/profit maximization under different θ assumptions.

3.2 Problem Formulations

The revenue r_b of a bundle b can be expressed in terms of the optimal price p_b, and the number of consumers who would adopt b, as shown in Equation 2. This number of consumers is a function $\mathcal{F}(p_b, W, \theta)$ of the price charged p_b, consumers' willingness to pay for b (determined by the willingness to pay for items W, and the bundling coefficient θ). The definition of this function will be discussed in Section 4.

$$r_b = \max_{p_b \in \mathbb{R}^+} \; p_b \times \mathcal{F}(p_b, W, \theta) \tag{2}$$

We now define the *k-sized bundle configuration problem*. There are two instances corresponding to pure bundling and mixed bundling. In terms of notation, a size-k bundle refers to a bundle b of exactly $|b| = k$ component items. A k-sized bundle refers to a bundle b of size $1 \leq |b| \leq k$.

PROBLEM 1 (k-SIZED PURE BUNDLING). *Given W, θ, and an integer $k \geq 1$, find the* bundle configuration $\mathcal{X}_I$, *containing k-sized bundles meeting the following conditions:*

1. $\bigcup \mathcal{X}_I = I$, *i.e., the union of sets in $\mathcal{X}_I$ is I*
2. $\forall b_1, b_2 \in \mathcal{X}_I$, $b_1 \cap b_2 \neq \emptyset$ *implies* $b_1 = b_2$
3. $\sum_{b \in \mathcal{X}_I} r_b$ *is maximized, i.e., there is no other partitioning of I with a higher overall revenue.*

$$\mathcal{X}_I = \underset{\mathcal{X} \text{ is a configuration of } I}{\operatorname{argmax}} \sum_{b \in \mathcal{X}} r_b \tag{3}$$

The bundle configuration that meets the above conditions is called *optimal*. The parameter k limits the maximum size of the bundles. For information goods (e.g., cable television), bundle sizes can grow very large, e.g., hundreds [3]. For physical goods (e.g., books), smaller bundle sizes may be more appropriate. The first condition specifies how the collection of all bundles should make up the full set of items. The second condition characterizes the pure bundling strategy, by requiring that $\mathcal{X}_I$ is a strict partition of I, i.e., no overlap between bundles. This prevents having both a bundle, as well as its component items both available. For instance, a cable TV provider (e.g., Starhub, SingTel) may partition a large number of cable TV channels into a small number of non-overlapping bundles. There may be other scenarios requiring overlapping bundles, and such scenarios are out of the scope of the current work. Finally, the last condition specifies the revenue maximization objective.

Mixed bundling has the distinction of allowing both a bundle and its component items being available. This is done by the subsumptions in the second condition in Problem 2.

PROBLEM 2 (k-SIZED MIXED BUNDLING). *Given W, θ, and an integer $k \geq 1$, find the* bundle configuration $\mathcal{X}_I$, *containing k-sized bundles meeting the following conditions:*

1. $\bigcup \mathcal{X}_I = I$, *i.e., the union of sets in $\mathcal{X}_I$ is I*
2. $\forall b_1, b_2 \in \mathcal{X}_I$, $b_1 \cap b_2 \neq \emptyset$ *implies* $b_1 \subseteq b_2$ *or* $b_2 \subseteq b_1$
3. $\sum_{b \in \mathcal{X}_I} r_b$ *is maximized, i.e., there is no other partitioning of I with a higher overall revenue.*

$$\mathcal{X}_I = \underset{\mathcal{X} \text{ is a configuration of } I}{\operatorname{argmax}} \sum_{b \in \mathcal{X}} r_b \tag{4}$$

Assumptions. We list some assumptions in this paper, which are conventionally used in the bundling literature [3].

- *Single Price:* Each unique bundle has one price, i.e., the seller demands the same price from consumers.
- *Single Unit:* Each consumer demands 0 or 1 unit of an item. This assumption can be relaxed if the number of units demanded is specified in the data.
- *No Budget Constraint:* Consumers are able to purchase any item, given sufficient willingness to pay.
- *No Supply Constraint:* Seller can sell to any number of consumers. For information goods, the marginal cost of providing an item to one more consumer is very low.

Proposed Approach. Our objective to arrive at the optimal bundle configuration can be decomposed into two sub-problems dealing with specific research issues. In Section 4, for a single bundle b, we seek to find its maximum revenue r_b. This requires us to model how a consumer may adopt a bundle at a particular price, and how to arrive at the price that maximizes the revenue. In Section 5, given a set of candidate bundles $\{b\}$ defined over I, and their corresponding maximum revenues $\{r_b\}$, we seek to find the configuration $\mathcal{X}_I$ that results in the maximum revenue.

4. SINGLE BUNDLE

Conventionally, a consumer adopts an item if the consumer's willingness to pay *equals or exceeds* the price [1]. We adopt this convention, but relax it with stochastic perturbations. We associate a random variable $\nu_{u,b}$ with a consumer u and a bundle b, i.e., $\nu_{u,b} = 1$ (u adopting b) and $\nu_{u,b} = 0$ (u not adopting b). $\mathrm{P}(\nu_{u,b} = 1 | p_b, w_{u,b})$ denotes the probability that u will adopt b, conditioned on the price p_b and u's willingness to pay $w_{u,b}$. This probability is higher for higher $w_{u,b}$ or lower p_b. Therefore, the expected number of consumers who will adopt b is as shown in Equation 5.

$$\mathcal{F}(p_b, W, \theta) = \sum_{u \in U} \mathrm{P}(\nu_{u,b} = 1 | p_b, w_{u,b}) \tag{5}$$

Together, Equation 2 and Equation 5 give the maximum revenue r_b of a bundle b. We first describe the modeling of $\mathrm{P}(\nu_{u,b} = 1 | p_b, w_{u,b})$, and then discuss how to estimate the optimal price p_b that results in the maximum r_b efficiently.

4.1 Modeling Adoptions Stochastically

To model $\mathrm{P}(\nu_{u,b} = 1 | p_b, w_{u,b})$, a suitable probability function is the sigmoid function, which is commonly used to model binary outcomes. The probability $\mathrm{P}(\nu_{u,b} = 1 | p_b, w_{u,b})$ can thus be expressed as in Equation 6. Parameters γ, α, and ϵ can be tuned to reflect different adoption decisions.

$$\mathrm{P}(\nu_{u,b} = 1 | p_b, w_{u,b}) = \frac{1}{1 + \exp\{-\gamma(\alpha \cdot w_{u,b} - p_b + \epsilon)\}} \tag{6}$$

The original sigmoid has $\gamma = 1$, $\alpha = 1$, $\epsilon = 0$. The red curve in Figure 1(a) shows how the adoption probabilities vary with the price p_b for a consumer with $w_{u,b} = 10$. At $p_b = 10$, the willingness to pay is exactly the same as the price, and the probability of adoption is 0.5. As price decreases, i.e., $p_b \to 0$, the probability of adoption increases. As price increases, i.e., $p_b \to \infty$, the probability decreases.

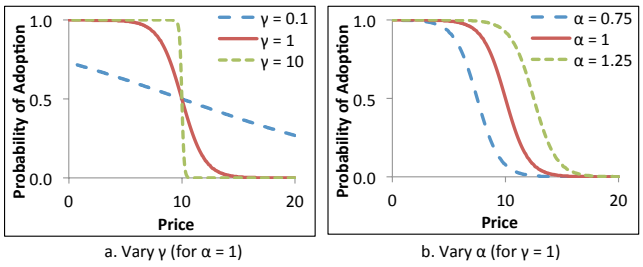

Figure 1: Modeling Probability of Adoption

Sensitivity to Price. To model consumers' sensitivity to price, the shape of the curve is parameterized by γ. This is illustrated by Figure 1(a), which contrasts $\gamma = 1$ with $\gamma = 10$ and $\gamma = 0.1$. As we decrease $\gamma < 1$, the curve gets more linear, representing a reduced sensitivity to price in making adoption decisions. As we increase $\gamma > 1$, the curve gets steeper. When $\gamma \to \infty$, we get the step function as a special case. To effect a step function's instant increase from 0 to 1, we add a small noise and set $\epsilon = 10^{-6}$.

Bias for Adoption. To model consumers' bias towards (or against) adoption, the curve is parameterized by α. This is illustrated by Figure 1(b), which contrasts $\alpha = 1$ with $\alpha = 0.75$ and $\alpha = 1.25$. As we decrease $\alpha < 1$, the curve shifts to the left, representing bias against adoption. As we increase $\alpha > 1$, there is a bias towards adoption, as indicated by the greater probability at any price point.

4.2 Determining Price and Revenue

Discretized Price Levels. The search for the optimal price involves investigating each candidate price p_b. In real-life scenarios, the seller would have a price list of T price levels. At worst, the price levels are the smallest atomic unit (e.g., cents). More commonly, it varies by larger increments.

Given such a price list, we associate each price level with a bucket and place each of the M consumers into a bucket according to her willingness to pay $w_{u,b}$. Identifying the bucket can be done through hashing (if equi-distanced price levels), or binary search (if arbitrary price levels). The optimal price level is determined by iterating through the T buckets. The number of buckets is usually fixed as a constant, and does not grow with the problem size. For experiments, we use 100 buckets, as we find that larger numbers do not result in much higher revenue. The complexity of pricing is $O(M)$.

Pure vs. Mixed Bundling. The above applies both to a bundle and to a component. For *pure bundling*, only a bundle or its components are on offer. For *mixed bundling*, a consumer's response to a bundle depends not only on the bundle's price, but also on the prices of its components. For example, u_1 may be willing to pay $w_{u_1,A} = \$12$ and $w_{u_1,B} = \$4$. Correspondingly, we have $w_{u_1,AB} = \$15.20$ (assuming $\theta = -0.05$). Suppose that mixed bundling offers the following prices: $p_A = \$8$, $p_B = \$8$, and $p_{AB} = \$15.20$. One may think u_1 would purchase the bundle, because $w_{u_1,AB} \geq p_{AB}$. That would be a counter-intuitive outcome [1]. u_1 could purchase A alone for \$8. To "upgrade" from A to the bundle $\{A, B\}$ would incur an implicit price for B of $p_{AB} - p_A = \$7.20$, more than u_1's willingness to pay for B (\$4). Therefore, u_1 would purchase item A alone. In an alternative scenario where the offer is: $p_A = \$12$, $p_B = \$4$, and $p_{AB} = \$15.20$, u_1 would purchase the bundle instead.

In other words, for pure bundling, the adoption decision is determined based on whether $p_{AB} \leq w_{u_1,AB}$, which means that the pricing of a bundle and its components can be done independently. In contrast, for mixed bundling, it is based on whether $p_{AB} - p_A \leq w_{u_1,B}$ and $p_{AB} - p_B \leq w_{u_1,A}$. This induces dependencies between prices of a bundle (p_{AB}) and its components (p_A and p_B). In most application scenarios, we expect that components are by default on offer, and the seller seeks greater revenue through bundling. Therefore, for mixed bundling, we adopt an *incremental* policy where the prices of components are determined first, and the price of a bundle is conditioned on the prices of components. We would investigate a relaxation of this policy as future work.

We apply the usual constraints for mixed bundling [18] to ensure a bundle would be a viable alternative to its components. First, $\forall i \in b,\ p_b > p_i$, i.e., the bundle price must be higher than any of its component's. Second, $p_b < \sum_{i \in b} p_i$, i.e., the bundle price is lower than the sum of its components' prices. These are not applicable to pure bundling.

5. BUNDLE CONFIGURATION

In the following, we first address the 2-sized bundle configuration problem, before addressing the case of $k \geq 3$.

5.1 Optimal Solution for 2-sized Bundling

For $k = 2$, the bundle configuration may contain bundles of size 1 or 2. There are N candidates of the former, and $N(N-1)/2$ candidates of the latter. Out of these candidate bundles, we need to select a subset of them to make up the bundle configuration $\mathcal{X}_I$ with the highest revenue.

First, we describe the *pure bundling* strategy. Bundles in $\mathcal{X}_I$ do not overlap with each other (see Section 3.2). If we model each item as a node in a graph (i.e., each bundle of two as an edge between two nodes, and each bundle of one as an edge from a node to itself), a valid bundle configuration is a set of edges, such that no two edges are incident on the same node and these edges collectively need to cover all the nodes, i.e., non-overlapping and no overage. In graph theory, such a collection of edges are known as a graph matching.

2-sized bundling can thus be reduced to graph matching. We construct a "complete" graph $G(V, E)$, where each vertex in V corresponds to an item in I, i.e., $|V| = N$. The set of edges E contains $N + N(N-1)/2$ edges. There is one edge from a vertex to itself (a candidate bundle of size 1). There is also an edge between any pair of vertices (a candidate bundle of size 2). Every edge is weighted by the maximum revenue of the item or bundle (see Section 4). It is easy to see that the optimal 2-sized bundling configuration is a maximum weight matching in G, with the highest sum of edge weights (total revenue). The advantage of this formulation is the existence of polynomial time algorithms for maximum weight matching, e.g., the Edmonds algorithm [14], with a complexity of $O(|E||V|^{\frac{1}{2}})$. For experiments, we use the LEMON library (`http://lemon.cs.elte.hu/trac/lemon`). To compute the edge weights (Section 4.2), we need to scan the M users once for each edge, which takes $O(MN^2)$. The complexity of this algorithm is thus $O(MN^2 + N^{2.5})$.

For *mixed bundling*, a similar formulation applies, with an adjustment whereby the edge weight between two different vertices is the maximum revenue expected from offering both a bundle and its two components (see Section 4.2). A bundle is feasible if offering both the bundle and its components bring in more revenue than offering its components alone.

5.2 Complexity of k-sized Bundling

Complexity. The case for $k \geq 3$ is more complex. Finding the optimal selection of potential bundles is intractable.

THEOREM 1. *3-sized pure bundling problem is NP-hard.*

PROOF SKETCH. 3-sized pure bundling problem can be expressed in terms of finding a maximum matching in a hypergraph containing edges of size-1, size-2, and size-3. We now show that 3-sized pure bundling problem is NP-hard through a reduction from the maximum 3-uniform hypergraph matching (known to be NP-hard [20]).

For any instance of maximum 3-uniform hypergraph matching problem for a hypergraph H, we can construct an instance of maximum hypergraph matching problem for H', where H' is not necessarily 3-uniform. H' is created as follows. For each original edge in H, we create an edge in H' with the weight $3 + \Delta$, where Δ is a fixed positive constant. In addition, we add "dummy" edges of size-1 (weight 1), size-2 (weight 2), and size-3 (weight 3) to H' that are not already in H. S' is a maximum matching in H', if and only if S is a maximum matching in H. Since finding the maximum matching S in H is NP-hard, finding the solution S' in H' (the 3-sized pure bundling problem) is also NP-hard. □

Because pure bundling is a special case of mixed bundling where a bundle and its components are not allowed to coexist, mixed bundling is likely as complex as, if not more complex than pure bundling, because a mixed bundling configuration may contain both a bundle and its components.

Connection to Weighted Set Packing. If we first enumerate all the $K = 2^N - 1$ potential bundles, the problem of pure bundling configuration among these potential bundles can be reduced to an instance of weighted set packing [9], for which there exists a solution with a known approximation bound. In weighted set packing, the input consists of K sets, where each set b_j has a known weight w_j. The objective is to find a collection of sets (that are pairwise disjoint) resulting in the highest aggregate weight. In our case, a set is a candidate bundle, and its weight is its revenue.

Optimal Solution. The optimal solution to weighted set packing, and thus to pure bundling problem, can be computed using the following Integer Linear Program (ILP). For every potential bundle b_j, we associate it with a binary variable x_j, which takes the value of 1 if b_j is selected as part of the solution, and 0 otherwise. Each b_j is also associated with a real-valued positive weight w_j, which represents its revenue. The objective is to determine x_j's to maximize the revenue using a combination of non-overlapping bundles.

$$\text{maximize} \sum_{j=1}^{K} x_j \times w_j$$

$$\text{subject to} \sum_{b_j:\ i \in b_j} x_j \leq 1, \text{for all } i = 1, \ldots, N$$

$$x_j = \{0, 1\}, \text{for all } j = 1, \ldots, K$$

This program is intractable for large N, which generates $K = 2^N - 1$ candidate bundles. In turn, the ILP has to find a solution within a space of $2^K - 1$ possible settings of x_j's. In practice, it is computable only for very small problem sizes. For experiments, we use the Gurobi ILP solver (`http://www.gurobi.com/`).

Approximable Solution. Because weighted set packing itself is also NP-hard, there exists approximation algorithms. The current best known solution is a greedy approach that repeatedly selects the next set with the highest average weight per item and removes other sets that overlap with the selected sets from future consideration. For N items, this approach is guaranteed to produce a solution within a factor of $\sqrt{N}$ less than the optimal solution [15].

Though this connection to weighted set packing allows us to establish the approximability of the problem, in practice existing weighted set packing solutions are still intractable for our scenario due to the requirement of enumerating and computing the revenues of all possible candidate bundles beforehand, a step that by itself has an $O(M \cdot 2^N)$ complexity. This is in addition to the complexity of weighted set packing algorithm (ILP or greedy). Therefore, we develop more efficient heuristic algorithms below, which do not require prior enumeration of all subsets. We will compare them experimentally to the weighted set packing solutions in Section 6.

5.3 Heuristic Solutions for k-sized Bundling

Here, we propose two heuristic algorithms for $k \geq 3$. Both algorithms can be applied to pure and mixed bundling. For clarity, we would first describe pure bundling, and highlight mixed bundling's differences in Section 5.3.3.

5.3.1 Matching-based Algorithm

Since the problem is tractable for $k = 2$ but not for $k \geq 3$, instead of solving it directly for k-sized bundles, one promising approach is to iteratively construct ever larger bundles. The pseudocode of this matching-based algorithm is given in Algorithm 1. We describe the *pure bundling* strategy here. For now, please ignore the lines specific to mixed bundling.

Each iteration is concerned with finding the best bundle configuration from the current components. The same Edmonds algorithm can be used in each iteration. For example, in the first iteration, we form size-2 bundles from combinations of size-1 bundles. In the second iteration, we treat these size-2 bundles as if they are singular items, and create another instance of 2-bundling problem with the size-1 and size-2 bundles as initial bundles, allowing a size-3 bundle to be created from a combination of a size-1 bundle and a size-2 bundle, or a size-4 bundle to be created from a combination of two size-2 bundles. This continues till we reach the maximum size k, or until there is no more gain in revenue.

We apply two pruning strategies to improve efficiency. In the first iteration, instead of all possible size-2 bundles, we only consider pairs of items for which at least one customer has non-zero willingness to pay for both. We cannot extract any positive remaining willingness to pay for the second item from customers who each want to buy only one item.

In subsequent iterations, when forming the edges in the graph, other than self-loop edges (the status quo), the other pruning strategy is to only introduce a new edge involving at least one newly-formed vertex in the current iteration. Edges in previous iterations that do not form a collapsible vertex will never form a bundle in the subsequent iterations as they are not favored over their components. Because of this "diminishing" effect of bundling, the number of iterations will effectively be bounded, as analyzed below.

At the start, there are N vertices and in the order of N^2 edges. For any given iteration, an edge between two vertices is either merged (if selected in the matching) or deleted. In the worst case, all non self-loop edges will be selected by the matching. Thus, the maximum number of edges, $|E|$,

Algorithm 1: Matching-based Algorithm

Initialize $\mathcal{X}_I$ to be a set of size-1 bundles.
Initialize $\mathcal{X}'_I$ to be an empty set.
Initialize R with the revenue of components.
while *true* **do**
  Construct a graph G with $\mathcal{X}_I$ as vertices.
  Populate G with edges involving newly-formed bundles.
  Compute the weight of each edge (see Section 4.2).
  Obtain the maximum weight matching S in G.
  Compute R', the weight or revenue of S.
  if $R' \leq R$ **then**
    Break.
  $R \leftarrow R'$
  for *each selected edge in* S **do**
    Remove the edge's vertices from $\mathcal{X}_I$.
    Collapse the edge into a new vertex in $\mathcal{X}_I$.
    if *mixed bundling* **then**
      Insert the edge's vertices into $\mathcal{X}'_I$.
Return $\mathcal{X}_I \cup \mathcal{X}'_I$.

Algorithm 2: Greedy Algorithm

Initialize $\mathcal{X}_I$ to be a set of size-1 bundles.
Initialize $\mathcal{X}'_I$ to be an empty set.
Initialize R with the revenue of components.
while *true* **do**
  for *every pair of elements* $b_1, b_2 \in \mathcal{X}_I$ **do**
    Form a candidate bundle $b' = b_1 \cup b_2$ of size $\leq k$.
    Compute the absolute gain in revenue:
    $r_\Delta = r_{b'} - r_{b_1} - r_{b_2}$ (see Section 4.2).
  Let $b' = b_1 \cup b_2$ be the candidate bundle with highest absolute revenue gain r_Δ.
  if $r_\Delta \leq 0$ **then**
    Break.
  $\mathcal{X}_I \leftarrow \mathcal{X}_I - \{b_1, b_2\}$
  $\mathcal{X}_I \leftarrow \mathcal{X}_I \cup b'$
  $R \leftarrow R + r_\Delta$
  if *mixed bundling* **then**
    $\mathcal{X}'_I \leftarrow \mathcal{X}'_I \cup \{b_1, b_2\}$
Return $\mathcal{X}_I \cup \mathcal{X}'_I$.

is no more than $(\frac{N}{2})^2$ after one iteration, $(\frac{N}{4})^2$ after two iterations, and so on. Taking into account the Edmonds algorithm's complexity of $O(|E||V|^{\frac{1}{2}})$, this is a geometric series of the form $a + ar + ar^2 + ...$, with $a = N^{2.5}$ and $r = \frac{1}{2}^{2.5}$, whose summation is bounded by $\frac{a}{1-r} = \frac{N^{2.5}}{1-\frac{1}{2}^{2.5}}$. Since the denominator is a constant, the complexity of matching across iterations is effectively still $O(N^{2.5})$.

To compute the edge weights, we need to scan the database of M users' willingness to pay once in every iteration, and update the revenue computation of all newly-formed bundles. Because the number of edges $|E|$ diminishes in a geometric series as above, the revenue computation requires $O(MN^2)$. The complexity of this algorithm is $O(MN^2 + N^{2.5})$. Realistically, the number of items N to be bundled is not extremely large. The number of users M may in some cases be large but it may be sufficient to take a significant sample of users, rather than using the data of all users.

5.3.2 Greedy Algorithm

The above matching-based approach is oriented towards finding the best configuration globally across all bundles in the partition $\mathcal{X}_I$. An alternative approach is to find only *one* best new bundle in each iteration. This new bundle can then immediately participate in the selection of the next bundle.

Algorithm 2 encapsulates this approach. In each iteration, it tries to perform a merging operation involving two existing bundles that result in the highest absolute gain in revenue over the component bundles. This newly merged bundle then participates in the next iteration searching for the next best merged bundle. This continues until a stopping condition is met. One natural stopping condition, which we adopt in this paper, is when there is no more revenue gain. An alternative stopping condition is to continue anyway till there is only a single bundle of N items, and then traversing all previous solutions to find the one with the maximum revenue. Empirically, this would increase running time significantly without producing meaningful revenue gain.

The algorithm may take up to N outer iterations, because in each iteration, the number of bundles in the configuration reduces by exactly 1, by collapsing two existing bundles into a new bundle. The first iteration involves $O(N^2)$ revenue computations, as the revenues of all candidate bundles of size-1 and size-2 need to be computed. In each subsequent iteration, we only need up to N revenue computations, involving the new bundle with an existing bundle. Each revenue computation itself will need $O(M)$ (see Section 4.2). There is also the cost of picking the bundle with the maximum gain in each iteration, which differs in complexity depending on the specific implementation (e.g., priority queue involves $O(\log N)$ per insertion/removal). Therefore, the overall complexity is approximately $O(MN^2 + N^2 \log N))$.

5.3.3 Pure Bundling vs. Mixed Bundling

The above matching-based and greedy algorithms are applicable for both bundling strategies (pure and mixed), with several differences. For one thing, the key difference between the two is how the revenue of a bundle is computed (see Section 4.2). Beyond that, another difference, as shown in Algorithm 1 and Algorithm 2, is that for mixed bundling, we retain in $\mathcal{X}'_I$ the subset of bundles replaced in previous iterations. These represent components that are also available to consumers, in addition to the subsuming bundles, which are finally returned as outputs of the algorithms. In contrast, for pure bundling, $\mathcal{X}'_I$ remains an empty set throughout.

6. EXPERIMENTS

Our objectives are two-fold. First, we seek to evaluate the effectiveness of our algorithms as compared to tractable non-bundling and bundling baselines. Second, we also seek to evaluate the efficiency of our algorithms through a scalability study and a comparison to weighted set packing solutions that attempt to produce optimal or approximable solutions.

6.1 Experimental Setup

6.1.1 Data

The willingness to pay matrix W is difficult to obtain [29]. While a small amount of data may be available to commercial organizations, there is no such publicly available dataset for research. We observe that the wealth of online ratings contain important signals about users' stated preferences, and simulate willingness to pay based on these ratings.

We use the UIC dataset of ratings crawled from Amazon.com [21]. Each rating is assigned by a user to a product,

on a scale of 1 (lowest) to 5 (highest). From this dataset, we extract the largest category of products, which is *Books*. Since the ratings of some users or some books are very sparse, we iteratively remove users and items with less than ten ratings until all users and items have ten ratings each. After this pre-processing, we have 4,449 users, 5,028 items (books), and 108,291 ratings. The distribution of ratings is as follows: 3% are ratings of 1, followed by 5%, 13%, 29%, and finally 49% are ratings of 5. In terms of the number of items, this dataset size is comparable or even larger than most real-life needs. This is because realistically, not all items in a seller's inventory are up for bundling.

A consumer's willingness to pay depends on a number of factors, including her need or preference for that item, as well as the value of the item. We assume that the former is partially indicated by her rating on the item, whereas the latter is partially indicated by the item's sales price. In addition to ratings, the sales price is also known in the dataset. The distribution of items by price is as follows: 50% of items have a sales price below \$10, 45% are between \$10-\$20, and a small fraction (4%) are above \$20.

We assume a linear relationship between ratings and willingness to pay [11, 22], and propose the following function. Suppose that p is the listed sales price of an item at Amazon. Presumably, some consumers would have willingness to pay below p, and some above p. We assume that the *highest* willingness to pay of any consumer for that item will be $\lambda \times p$, for some conversion factor $\lambda \geq 1$. We equate the highest possible rating r_{max} to $\lambda \times p$. For any other rating $r < r_{max}$, the corresponding willingness to pay is $\frac{r}{r_{max}} \times \lambda \times p$. For example, if $\lambda = 1.25$ and the listed price for an item is $p = \$10$, a consumer who rated 5 stars is considered willing to pay $\frac{5}{5} \times 1.25 \times \$10 = \$12.50$ for the item. Ratings of 4, 3, 2, and 1 map to willingness to pay \$10, \$7.50, \$5, and \$2.50.

6.1.2 Metrics

To evaluate the effectiveness of an algorithm in terms of the revenue maximization objective, we rely on two metrics.

Revenue Coverage. One informative measure is how close we get to the absolute maximum. The aggregate willingness to pay in the input matrix W is effectively the upper bound of revenue. The *revenue coverage* of an algorithm is the ratio (expressed as percentage) of its revenue to the total amount of willingness to pay. For example, if an algorithm generates \$11 of revenue, while the total willingness to pay (of all users on all items) is \$20, the revenue coverage is $\frac{\$11}{\$20} = 55\%$. The "perfect" score would be 100%.

Revenue Gain. Another measure is how much an algorithm gains over the *Components*. This quantifies the direct benefit of bundling, as opposed to selling individual items. The *revenue gain* of an algorithm is the fractional gain (expressed as percentage) over the revenue of *Components*. For example, if an algorithm generates \$11 of revenue, while *Components* generates \$10, then the revenue gain is $\frac{(\$11-\$10)}{\$10} = 10\%$. A good algorithm is expected to have positive gain. The higher the gain, the better.

6.1.3 Comparative Methods

Our Methods. We compare four versions of our approach. We have two problem formulations (pure bundling and mixed bundling), and two algorithms (matching and greedy), yielding four algorithms, namely: *Pure_Matching*, *Pure_Greedy*, *Mixed_Matching*, and *Mixed_Greedy*.

λ	Optimal pricing	Amazon's pricing
1.00	77.7%	59.0%
1.25	77.7%	75.1%
1.50	77.7%	62.6%
1.75	77.7%	62.8%
2.00	77.7%	54.9%

Table 2: Revenue Coverage at Different λ's

Notation	Description	Default Value
λ	conversion factor	1.25
θ	bundling coefficient for willingness to pay	0
k	max size for a bundle	∞ (no size limit)
γ	stochastic sensitivity to price	10^6 (step function)
α	stochastic bias for adoption	1 (unbiased)

Table 3: Default Parameter Settings

We compare our methods to two categories of baselines: methods selling individual items, and bundling baselines.

Non-Bundling Baseline. For *Components*, there are two options, whether to determine the optimal pricing (see Section 4) or to take the sales price in the Amazon dataset. In Table 2, we compare the revenue coverages resulting from these two options for different λ. For the optimal pricing, the revenue coverage remains at around 77.7% across λ's. For Amazon's pricing, revenue coverages vary with λ, with the highest coverage of 75.1% at $\lambda = 1.25$. Optimal pricing is stronger baseline than Amazon's pricing. It has higher revenue coverage at any λ. It is sufficient to compare to optimal pricing, which is more difficult to beat. We use the optimal pricing for *Components*, and set $\lambda = 1.25$ for which Amazon's pricing comes closest to *Components* in revenue.

Bundling Baselines. There is a lack of suitable baseline, as prior work focuses mostly on only 2 items [1]. One common feature on Amazon and other online shops is "Frequently Bought Together". We simulate this using frequent itemsets [2]. We treat the Amazon dataset as transaction data. Each transaction corresponds to a consumer, containing the items for which this consumer has non-zero willingness to pay. We mine frequent itemsets using MAFIA [8].

With the frequent itemsets as candidate bundles, we construct a bundle configuration by greedily picking the itemset with the highest absolute gain in revenue over its components. We remove itemsets that overlap with the selected itemsets, and iteratively pick the next best itemset until we cover all the items. Individual items are used as candidates even if they do not meet the minimum support (this favors the frequent itemset approach). We experimented with various minimum supports and found 0.1% to produce the highest revenue. We include two baselines for pure and mixed bundling, i.e., *Pure_FreqItemset* and *Mixed_FreqItemset*.

6.2 Comparison against Baselines

We organize this section by the parameters involved in our approach. The default settings are given in Table 3.

Bundling Coefficient θ. Willingness to pay for bundles is parameterized by θ (see Section 3.1). Figure 2 shows the effects of θ on various methods. There are two y-axes: the left is for revenue coverage, and the right is for revenue gain. Since θ only applies to bundling, *Components* is not affected by θ. Revenue gain, expressed relative to *Components*'s revenue, thus has the same trend as the revenue coverage. For negative θ (substitute), a consumer's willingness

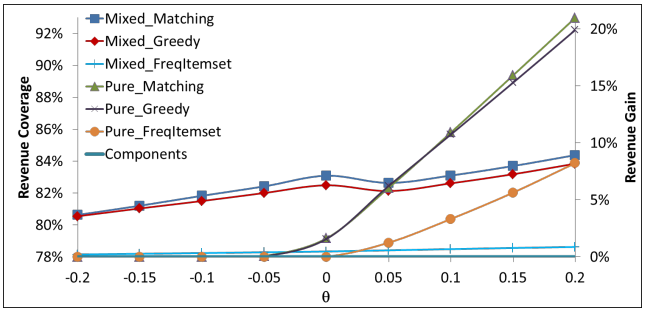

Figure 2: Experiments with different θ values

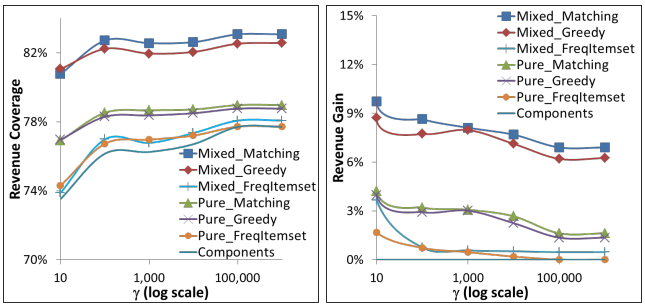

Figure 3: Experiments with different γ values

to pay for bundles is discounted, creating a tendency not to form bundles. *Components* has the lowest coverage at 77.7%. *Mixed_Matching* and *Mixed_Greedy* have the highest revenue coverages, because they can target the segments of the market who can afford bundles, while still making the components available to whose who cannot. *Pure_Matching* and *Pure_Greedy* are more sensitive to negative θ. As θ decreases, they degenerate into *Components*. The two bundling baselines *Mixed_FreqItemset* and *Pure_FreqItemset* also have very low revenue coverages not much different from *Components*.

For positive θ (complementary), a consumer's willingness to pay is augmented when items are bundled. *Pure_Matching* and *Pure_Greedy* are especially effective when $\theta \gg 0$. By making only the bundle available, the seller can extract a higher price and thus a higher revenue. *Mixed_Matching* and *Mixed_Greedy* also increase in revenue because of the higher willingness to pay for bundles, but not as steeply as pure bundling. The baselines underperform our corresponding methods. *Mixed_FreqItemset* is worse than *Mixed_Matching* or *Mixed_Greedy*. In turn, *Pure_FreqItemset* is worse than *Pure_Matching* or *Pure_Greedy*.

This shows that bundling is advantageous. The bundling methods do not go below *Components*, because they revert to *Components* if there is no better solution. Often, bundling results in higher revenues. Frequent itemsets are not optimized for revenue. By confining to frequent itemsets, we may miss out on candidate bundles with higher revenue gain.

Among our own methods, we see that the matching-based algorithms (*Mixed_Matching* and *Pure_Matching*) outperform their greedy counterparts (*Mixed_Greedy* and *Pure_Greedy*) slightly in terms of achieving a higher revenue coverage.

For subsequent experiments, we will set $\theta = 0$, which is also the conventional setting in the literature [1].

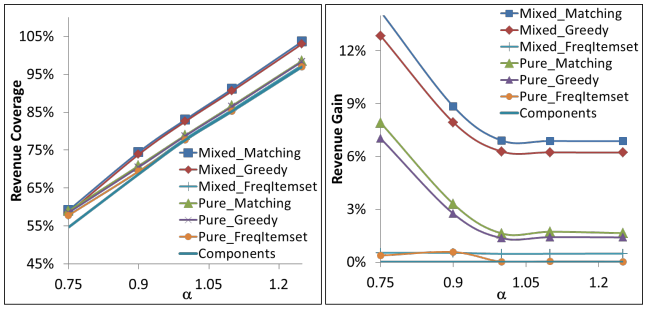

Figure 4: Experiments with different α values

We now look at factors affecting the number of adoptions of a bundle. Since this is stochastic, we average revenues across ten runs. Adoption is modelled stochastically (Equation 6 in Section 4.1), with parameters γ and α.

Stochastic Sensitivity γ. Figure 3(a) shows how revenue coverage varies with γ. As γ increases, revenue coverage also increases, though at a decreasing rate. Low γ indicates a lot of uncertainty whether a consumer will adopt a bundle. The price needs to be lower to compensate for this uncertainty. A high γ reduces this uncertainty, allowing the seller to set a higher price, and earns a higher revenue.

Figure 3(b) shows the trend of revenue gain, which decreases as γ increases. Since revenue gain represents performance relative to *Components*, this implies that bundling is more robust. Lower γ indicates greater uncertainty, requiring a greater reduction in price and a greater loss in revenue by *Components*. Bundling creates a flatter distribution of willingness to pay, stemming the reduction in revenue.

The relative standing of methods remains as before across γ's. As default, we set γ to a high value, i.e., 10^6 to simulate the step function (the convention in bundling literature [1]).

Stochastic Bias α. The parameter α indicates a bias for adoption, by shifting the sigmoid curve (Equation 6). Higher α results in a higher probability of adoption. The effects are similar to γ. Both Figure 4(a) for revenue coverage and Figure 4(b) for revenue gain show similar trends as in Figure 3. However, the increase is linear in Figure 4(a) as α can keep increasing the probability. The plateau in Figure 3(a) is because of reaching a step function in the limit. As default, we set $\alpha = 0$, i.e., no bias.

Size Constraint k. In previous experiments, the maximum size of bundles is unconstrained. Bundles can grow to any size as long as it produces some gain in revenue. We now investigate constraining the maximum bundle sizes by varying k. Because *Components* is not affected by k, the trend lines of revenue coverage and revenue gain are the same. We show both in Figure 5 using dual y-axes.

When $k = 1$, bundling is equivalent to the *Components*. For $k = 2$, bundling starts to gain over *Components*. As we increase $k \geq 3$, the revenue keeps growing though at a slower rate. Previous work focuses mostly on bundles of size 2, and obtaining the optimal bundling configuration of size 3 and above is NP-hard. This experiment shows that while bundles of size 2 could produce some revenue gain, there is still a significant additional amount of revenue to be gained from bundles of larger sizes. This validates our approach in designing heuristic algorithms to discover larger bundles.

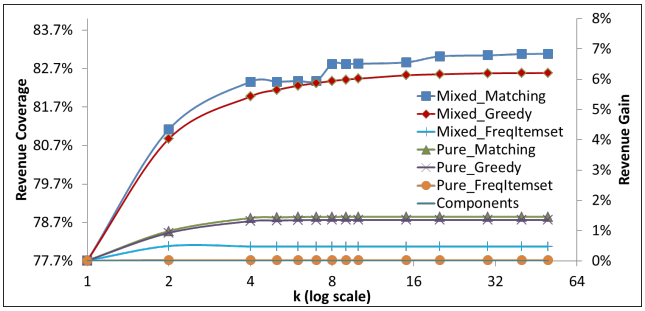

Figure 5: Experiments with different max size k

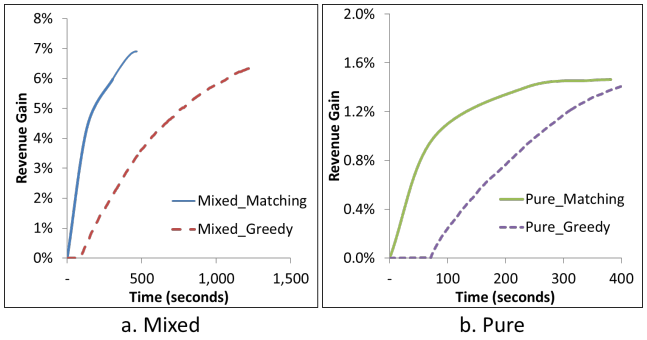

Figure 6: Revenue Coverage vs. Time

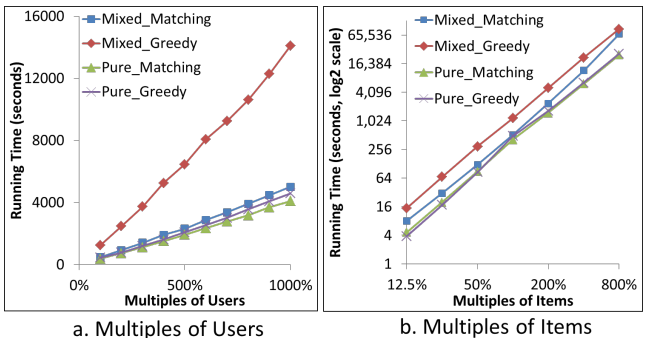

Figure 7: Scalability of Bundling Algorithms

6.3 Computational Complexity

We briefly comment on the efficiencies of our matching-based and greedy algorithms. Timing is based on an Intel Xeon E5-2667 2.9GHz machine with 70GB RAM.

Revenue Coverage vs. Time. For both the matching and greedy algorithms, each additional iteration seeks to increase revenue, at the cost of increased running time. For mixed bundling, Figure 6(a) shows that as the number of iterations increases, both revenue gain and running time increase. Overall, *Mixed_Matching* offers a better trade-off of revenue vs. time than *Mixed_Greedy* does. For the same revenue gain, *Mixed_Matching* is faster. For the same running time, *Mixed_Matching* has a higher revenue gain. In total, *Mixed_Matching* requires 10 iterations over 466 seconds till the revenue converges. In the first iteration, the revenue gain is 4.4%. Over the next iterations, revenue coverage further increases, reaching a total gain of 7%. For *Mixed_Greedy*, the increase is more gradual, over many more iterations (4347) and longer time (1241 seconds).

For pure bundling, Figure 6(b) shows as the number of iterations increases, both revenue gain and running time increase. The trends are similar to the mixed bundling strategy. The matching-based algorithm has much fewer iterations (6 vs. 2131) over a shorter period of time (382 vs. 449 seconds), as compared to the greedy algorithm. The former also presents a steeper increase in revenue gain over time, presenting a better trade-off of revenue gain vs. time.

Scalability. To investigate the scalability of the proposed algorithms, we measure how their running times are affected by the number of users, as well as by the number of items.

To create larger datasets with the same number of items, but varying number of users, we clone the users in the same dataset using a multiplication factor. For example, the original dataset has a factor of 100%. For the factor of 200%, we have the same number of items, and twice as many users (with the same ratings as the original users). Figure 7(a) shows how running time varies with different multiplication factors. All the algorithms scale linearly with the number of users, which is reasonable since it only affects searching for the optimal price, which is $O(M)$ (see Section 4.2).

Figure 7(b) shows the scalability with respect to different multiples of items. Both the x- and y-axes are in $\log_2$ scale. The linear growth in log-log axes indicates that the growth of running time is polynomial to the number of items, which is consistent with the complexity analysis in Section 5.

The matching algorithms are more efficient than the greedy algorithms, because greedy requires more iterations to converge, since in each iteration it only creates one bundle.

6.4 Comparison to Weighted Set Packing

We now compare to the weighted set packing solutions described in Section 5.2, i.e., the ILP-based optimal solution (*Optimal*) as well as the greedy solution with known approximation bound (*Greedy_WSP*). Our objective is to show that our tractable algorithms can reach close to the optimal solution, and yet with greater efficiency. The following comparison includes only pure bundling as the reduction to weighted set packing is only defined for pure bundling.

Because weighted set packing requires enumeration of all subsets of items, it is only tractable for small problem sizes. To produce a small-scale dataset, we randomly select $N \in \{10, 15, 20, 25\}$ items from the universal set of 5,028 items, but include all the users. Even for this small-scale datasets, the enumeration and revenue computation for $2^N - 1$ subsets of items require 0.8 seconds for 10 items, 32 seconds for 15 items, 24 minutes for 20 items, and 15 hours for 25 items. The $2^N - 1$ is just the number of potential bundles, and finding a solution, i.e., a set of pair-wise non-overlapping bundles covering all items and having the maximum profit, requires another exponential search in this space. It is prohibitive for any larger number of items, and is not practical in real scenarios.

Since our algorithms still produce an optimal solution for 2-sized bundling, to test the heuristics for $k \geq 3$, we retain only the samples resulting in at least one bundle of size 3 or larger. We average the results across 10 random samples.

Comparison to Optimal. Table 4 compares our algorithms *Pure_Matching* and *Pure_Greedy* (using settings in Table 3) to the weighted set packing solutions in terms of revenue coverage. Notably, for sample sizes of 10 to 20 our algorithms reach the same revenue coverage as *Optimal* for

	Revenue Coverage (percent)			
	$N = 10$	$N = 15$	$N = 20$	$N = 25$
Pure_Matching	78.1%	77.8%	77.9%	77.2%
Pure_Greedy	78.1%	77.8%	77.9%	77.2%
Optimal	78.1%	77.8%	77.9%	-
Greedy_WSP	68.1%	65.2%	64.9%	64.3%

Table 4: Comparison to Weighted Set Packing: Revenue

	Running Time (seconds)			
	$N = 10$	$N = 15$	$N = 20$	$N = 25$
Pure_Matching	0.01	0.01	0.01	0.02
Pure_Greedy	0.07	0.10	0.13	0.16
Optimal	0.20	4.60	235.38	-
Greedy_WSP	0.02	0.49	24.71	706.28

Table 5: Comparison to Weighted Set Packing: Time

all the random samples. This could well be due to the relatively small sample sizes, but the *Optimal* solution can only be computed for small sample sizes.

Table 5 shows how our algorithms are much more efficient than *Optimal*. The running time of *Optimal* grows extremely fast. Importantly, this running time has not even included the time to enumerate all the subsets, which may be up to 15 hours for 25 items. Even so, the result for 25 items cannot be computed due to insufficient computing resources to process the $2^{25} - 1$ or 33 million boolean variables required by the ILP. Extending *Optimal* to even larger N is not feasible for the available computational resources.

Comparison to Greedy_WSP. Table 4 also shows that *Pure_Matching* and *Pure_Greedy* outperform the current approximation solution for weighted set packing *Greedy_WSP* (with known approximation factor of $\sqrt{N}$). Our methods have higher revenue coverages across different sample sizes from 10 to 25. Empirically, it is evident that our algorithms reach a better approximation of *Optimal* than *Greedy_WSP*.

This is achieved at greater efficiency as well. Table 5 shows that not only *Greedy_WSP* takes more time, but the running time also grows much faster. For $N = 25$, our algorithms complete in less than a second. *Greedy_WSP* requires more than ten minutes (which would be much worse if we include the 15 hours for the subset enumeration beforehand).

6.5 Case Study

To illustrate bundling, we now provide a small example case from the real data. Due to space limitation, we only showcase mixed bundling. Table 6 shows three books: *The Sands of Time*, *Two Little Lies*, and *Born in Fire*. The starting point is the prices when the items are sold individually. The computed optimal price is 7.99 for *The Sands of Time* netting 10 buyers, 6.99 for *Two Little Lies* netting 9 buyers, and *Born in Fire* netting 9 buyers. In mixed bundling, individual items are always available for sale.

Next, we consider whether to make any bundle of size 2 available in addition to individual items. Here, we find that bundling (*Two Little Lies*, *Born in Fire*) results in one additional buyer, who would not purchase individual items because her willingness to pay for each item is below the individual price, but would purchase the bundle because her aggregate willingness to pay for both items meet the bundle price of 11.20, netting an additional revenue of 11.20. In contrast, bundling (*The Sands of Time*, *Born in Fire*) only results in additional revenue of 5.92, and bundling (*The Sands of Time*, *Two Little Lies*) generates no additional revenue. Since the three bundles are "overlapping", we select (*Two Little Lies*, *Born in Fire*) for its highest additional revenue.

Bundle	Price	Add. buyers	Add. revenue	Selected?
The Sands of Time	7.99	10	79.90	✓
Two Little Lies	6.99	9	62.91	✓
Born in Fire	7.99	9	71.91	✓
(The Sands of Time, Two Little Lies)	14.97	0	0	
(The Sands of Time, Born in Fire)	13.91	1	5.92	
(Two Little Lies, Born in Fire)	11.20	1	11.20	✓
(The Sands of Time, Two Little Lies, Born in Fire)	13.91	1	5.92	✓

Table 6: Case Study: Mixed Bundling

Finally, we see that by further bundling (*Two Little Lies*, *Born in Fire*) and *The Sands of Time* into a bundle of size 3, we net an additional revenue of 5.92 from someone who previously would only purchase *Born in Fire* alone for 7.99, but now would purchase the bundle of 3 at the price of 13.91.

6.6 Summary

The experiments consistently produce these findings:

- Our proposed bundling methods outperform the baselines in terms of revenue coverage. Bundling outperforms, or at least equals, *Components*, because it reverts to *Components* if it cannot find a better solution. Our methods also outperform the corresponding bundling baselines based on frequent itemsets, i.e., *Mixed_FreqItemset* and *Pure_FreqItemset* for mixed and pure bundling respectively. The standing among comparative methods is consistent and evident from the Figures 2 to 5 in Section 6.2.

- Mixed bundling and pure bundling are different strategies. Each has its own advantage depending on the assumption about the complementarity among items in a bundle (i.e., θ). This is evident from Figure 2.

- For both pure and mixed bundling, the matching algorithm outperforms the greedy algorithm, in terms of obtaining a higher maximum revenue coverage, as well as doing so in less time, as shown in Figure 6.

- Both of our algorithms, the matching-based and greedy heuristic algorithms, are relatively efficient, with running times growing polynomially with respect to items and linearly with respect to users, as shown in Figure 7.

- Our algorithms provide tractable solutions for the bundle configuration problem. Compared to weighted set packing, our algorithms achieve the same level of revenue coverage as the optimal solution (*Optimal*), and outperform the greedy solution with known approximation bound (*Greedy_WSP*) on all the random samples tested. Meanwhile, our running times are much faster, even without taking into account the time to enumerate all the subsets of items needed by weighted set packing that effectively renders the weighted set packing solutions intractable.

7. CONCLUSION

We address the bundle configuration problem by mining willingness to pay data from user-generated ratings data. This is a major difference from the traditional business approach where explicit solicitation from consumers is frequently required, which does not work in large scale. Our objective is to determine the bundle configuration that maximizes the total revenue. We address two variants of this problem, based on the respective bundling strategies of pure bundling and mixed bundling. We show that the problem is NP-hard for pure bundling of size 3 or more, and introduce a couple of effective heuristic solutions based on graph matching and greedy selection respectively. Experimentation shows that our approach results in higher revenues than individual components as well as bundles based on frequent itemsets. Moreover, the matching-based algorithm outperforms the greedy algorithm in effectiveness and efficiency.

8. ACKNOWLEDGMENTS

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA). The work of the third author was partially done during a visit to SA Center for Big Data Research hosted in Renmin University of China. This Center is partially funded by a Chinese National 111 Project "Attracting International Talents in Data Engineering and Knowledge Engineering Research".

9. REFERENCES

[1] W. J. Adams and J. L. Yellen. Commodity bundling and the burden of monopoly. *The Quarterly Journal of Economics*, pages 475–498, 1976.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, pages 487–499, 1994.

[3] Y. Bakos and E. Brynjolfsson. Bundling and competition on the Internet. *Marketing Science*, pages 63–82, 2000.

[4] P. Balakrishnan and V. S. Jacob. Genetic algorithms for product design. *Management Science*, 42(8):1105–1117, 1996.

[5] M.-F. Balcan and A. Blum. Approximation algorithms and online mechanisms for item pricing. In *EC*, pages 29–35, 2006.

[6] S. Basu Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu. Constructing and exploring composite items. In *SIGMOD*, pages 843–854, 2010.

[7] C. Breidert, M. Hahsler, and T. Reutterer. A review of methods for measuring willingness-to-pay. *Innovative Marketing*, 2(4):8–32, 2006.

[8] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *ICDM*, pages 443–452, 2001.

[9] B. Chandra and M. Halldórsson. Greedy local improvement and weighted set packing approximation. In *SODA*, pages 169–176, 1999.

[10] C. S. Chu, P. Leslie, and A. Sorensen. Bundle-size pricing as an approximation to mixed bundling. *The American Economic Review*, 101(1):263–303, 2011.

[11] J. C.-I. Chuang and M. A. Sirbu. Network delivery of information goods: Optimal pricing of articles and subscriptions. *Internet Publishing and Beyond: The Economics of Digital Information and Intellectual Property*, pages 147–176, 2000.

[12] M. Das, G. Das, and V. Hristidis. Leveraging collaborative tagging for web item design. In *KDD*, pages 538–546, 2011.

[13] S. De Vries and R. V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003.

[14] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

[15] R. Gonen and D. Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *EC*, pages 13–20, 2000.

[16] P. E. Green and V. Srinivasan. Conjoint analysis in consumer research: issues and outlook. *Journal of Consumer Research*, 5(2):103–123, 1978.

[17] A. Grigoriev, J. Van Loon, M. Sviridenko, M. Uetz, and T. Vredeveld. Bundle pricing with comparable items. In *Algorithms–ESA*, pages 475–486. 2007.

[18] J. P. Guiltinan. The price bundling of services: A normative framework. *Journal of Marketing*, 51(2):74–85, 1987.

[19] W. Hanson and R. K. Martin. Optimal bundle pricing. *Management Science*, 36(2):155–174, 1990.

[20] H. Huang, P.-S. Loh, and B. Sudakov. The size of a hypergraph and its matching number. *Combinatorics, Probability & Computing*, 21(3):442–450, 2012.

[21] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, pages 219–230, 2008.

[22] J. O. Kephart, C. H. Brooks, R. Das, J. K. MacKie-Mason, R. Gazzale, and E. H. Durfee. Pricing information bundles in a dynamic environment. In *EC*, pages 180–190, 2001.

[23] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized travel package recommendation. In *ICDM*, pages 407–416, 2011.

[24] Y. Liu, W.-k. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In *UBDM*, pages 90–99, 2005.

[25] D. McFadden. The choice theory approach to market research. *Marketing Science*, 5(4):275–297, 1986.

[26] I. Mendez-Diaz, P. Zabala, F. Bonchi, C. Castillo, E. Feuerstein, and S. Amer-Yahia. Composite retrieval of diverse and complementary bundles. *TKDE*, pages 2662–2675, 2014.

[27] R. Venkatesh and W. Kamakura. Optimal bundling and pricing under a monopoly: Contrasting complements and substitutes from independently valued products. *Journal of Business*, 76(2):211–231, 2003.

[28] R. Venkatesh and V. Mahajan. The design and pricing of bundles: a review of normative guidelines and practical approaches. *Handbook of Pricing Research in Marketing*, pages 232–257, 2009.

[29] F. Voelckner. An empirical comparison of methods for measuring consumers' willingness to pay. *Marketing Letters*, 17(2):137–149, 2006.

[30] M. Xie, L. V. Lakshmanan, and P. T. Wood. Comprec-trip: A composite recommendation system for travel planning. In *ICDE*, pages 1352–1355, 2011.