Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Constrained Assortment Optimization Under the Paired Combinatorial Logit Model

Rohan Ghuge, Joseph Kwon, Viswanath Nagarajan, Adetee Sharma

To cite this article:
Rohan Ghuge, Joseph Kwon, Viswanath Nagarajan, Adetee Sharma (2022) Constrained Assortment Optimization Under the Paired Combinatorial Logit Model. Operations Research 70(2):786-804. https://doi.org/10.1287/opre.2021.2188

http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 70, No. 2, March–April 2022, pp. 786–804
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Contextual Areas

Constrained Assortment Optimization Under the Paired Combinatorial Logit Model

Rohan Ghuge,[a] Joseph Kwon,[a] Viswanath Nagarajan,[a] Adetee Sharma[a]

[a] Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109

Contact: rghuge@umich.edu (RG); jyjkwon@umich.edu (JK); viswa@umich.edu, https://orcid.org/0000-0002-9514-5581 (VN); adetees@umich.edu (AS)

Received: January 5, 2019
Revised: June 5, 2020; February 26, 2021; May 1, 2021
Accepted: July 27, 2021
Published Online in Articles in Advance: December 7, 2021

Area of Review: Operations and Supply Chains

https://doi.org/10.1287/opre.2021.2188

Abstract: We study the assortment optimization problem when customer choices are governed by the paired combinatorial logit model. We study unconstrained, cardinality-constrained, and knapsack-constrained versions of this problem, which are all known to be NP-hard. We design efficient algorithms that compute approximately optimal solutions, using a novel relation to the maximum directed cut problem and suitable linear-program rounding algorithms. We obtain a randomized polynomial time approximation scheme for the unconstrained version and performance guarantees of 50% and $\approx 50\%$ for the cardinality-constrained and knapsack-constrained versions, respectively. These bounds improve significantly over prior work. We also obtain a performance guarantee of 38.5% for the assortment problem under more general constraints, such as multidimensional knapsack (where products have multiple attributes and there is a knapsack constraint on each attribute) and partition constraints (where products are partitioned into groups and there is a limit on the number of products selected from each group). In addition, we implemented our algorithms and tested them on random instances available in prior literature. We compared our algorithms against an upper bound obtained using a linear program. Our average performance bounds for the unconstrained, cardinality-constrained, knapsack-constrained, and partition-constrained versions are over 99%, 99%, 96%, and 99%, respectively.

Funding: This research was supported in part by the National Science Foundation Division of Computing and Communication Foundations (CCF) [Grant CCF-1750127] and Division of Civil, Mechanical, and Manufacturing Innovation (CMMI) [Grant CMMI-1940766].
Supplemental Material: The online appendices are available at https://doi.org/10.1287/opre.2021.2188.

Keywords: assortment optimization • submodularity • linear and semidefinite programming

1. Introduction

A prevalent operational problem faced by many firms is to select an assortment of products to offer their customers in order to maximize profit. In order to make such a decision, one needs to understand and model customer behavior. Conventional approaches to modeling customer behavior assume that every customer has a product he or she is interested in buying. If that product is offered, the customer makes the purchase. If it's not offered, then the customer leaves without making any purchase. These approaches disregard substitution behavior that has been observed in customers. Customers may substitute their favorite product for another one that is available if it meets their needs. Discrete choice models have been widely used to combat this issue. These models capture the demand of a product based on the attributes of the product itself and the attributes of the other products in the offered assortment. Thus, they help in modeling the substitution behavior exhibited by customers and also capture relationships between different products.

In this paper, we study assortment optimization problems when customer choice behavior is modeled by the paired combinatorial logit (PCL) model. The most basic versions of assortment optimization are unconstrained, capacitated, and knapsack constrained. In the unconstrained version, we can offer any subset of products. In the capacitated version, there is a limit on the number of products that we can offer. In the knapsack constrained (also called space constrained) version, there is a size associated with each product and there is a constraint on the total size of offered products. These three versions were studied by Zhang et al. (2020) and Feldman (2017). We provide significantly improved approximation algorithms for all of these problems. In addition, we provide the first constant-factor approximation algorithms for PCL assortment optimization under other natural constraints,

such as partition and multidimensional knapsack constraints. These constraints are significantly harder to handle and have not been studied previously. In the partition version, products are partitioned into different categories and there is a limit on the number of selected products in each category. In the multidimensional-knapsack version, each product has multiple attributes (for example, weight, volume, length) and there is a constraint on the total size of each attribute.

The PCL model is a random utility model, similar to the widely used multinomial logit (MNL) and nested logit (NL) models (McFadden 1974). In these models, the customer has a random utility for each product (drawn from some joint distribution) and chooses the product with maximum utility. The MNL model assumes that utilities of different products are independent. The NL model generalizes MNL by partitioning products into nests. Products in different nests have independent utilities, and there is equal correlation between utilities of products within the same nest. In contrast, the PCL model allows for correlations between utilities of any pair of products. So the PCL model is preferable in situations where there are significant correlations between the utilities of different products, for example, in selecting travel modes (Bekhor and Prashker 1998). It turns out that the PCL model is more general than MNL. Although the PCL model does not directly generalize the NL model, it allows for more general correlation structure as noted above; we refer the interested reader to Zhang et al. (2020) for a more thorough discussion on this. Moreover, as discussed in Koppleman and Wen (2000), estimating the parameters for the PCL model is advantageous over NL because we do not need to search among numerous NL nesting structures.

Although assortment optimization problems in the MNL and NL models have been studied extensively under various constraints (see, e.g., Talluri and van Ryzin 2004, Davis et al. 2013, Gallego and Topaloglu 2014, Feldman and Topaloglu 2015), assortment optimization in the PCL model was initiated only recently by Zhang et al. (2020). Our work adds to this literature by addressing PCL assortment optimization under many types of constraints, as described next.

1.1. Our Contributions

First, we establish a novel connection from the PCL assortment optimization problem to the maximum directed cut (max-dicut) problem. Max-dicut is a fundamental combinatorial optimization problem that involves selecting a subset of nodes in a directed graph so as to maximize the weight of outgoing edges from the chosen subset. The max-dicut instances obtained through this connection contain *negative* weight edges, so existing approximation algorithms for max-dicut are not directly applicable. However, using properties of the PCL model, we show that every node in the max-dicut instance has either all-positive or all-negative arcs leaving it. This enables us to use and extend approximation algorithms for max-dicut (with nonnegative weights).

Second, we combine the above insight with the approximation framework of Zhang et al. (2020) and new linear program (LP) rounding algorithms to obtain improved approximation algorithms for the unconstrained, cardinality-constrained, and knapsack-constrained assortment problems. For the unconstrained version, we obtain a randomized 0.874− approximation guarantee via a semidefinite program (SDP) relaxation. For the cardinality and knapsack versions, we obtain approximation guarantees of 0.5 and 0.25, respectively, using a linear program relaxation and a rounding technique called pipage rounding. Such a rounding method was previously known (Ageev et al. 2001) for the max-dicut problem with an equality constraint on the number of vertices: we extend this method to the more general setting of a knapsack constraint. We thus improve previously known approximation guarantees for all the basic versions of PCL assortment optimization: unconstrained (from 0.79 to 0.874), cardinality constrained (from 0.25 to 0.5), and knapsack constrained (from 0.083 to 0.25). For the cardinality-constrained assortment problem, another advantage of our approach is that it only requires solving two LPs as opposed to multiple (up to n) LPs needed in the previous algorithm of Zhang et al. (2020).

Third, we present an alternative approximation framework that is based on binary search and the relation to max-dicut. Our framework is more general than the one presented by Zhang et al. (2020) in the sense that we can use *any* approximation algorithm for constrained max-dicut. We show that the assortment optimization problem under any constraint $\mathcal{C}$ admits an $(\alpha_{\mathcal{C}} - \epsilon)$-approximation algorithm where $\alpha_{\mathcal{C}}$ is the approximation guarantee for max-dicut under the same constraint $\mathcal{C}$ and $\epsilon > 0$ is any constant. As applications, we obtain (i) an improved $(0.5 - \epsilon)$-approximation algorithm for the knapsack-constrained assortment problem and (ii) the first constant-factor approximation algorithms for the assortment problem under matroid and multidimensional knapsack constraints. The first application relies on an improved $(0.5 - \epsilon)$-approximation algorithm for max-dicut under a knapsack constraint that we obtain in this paper; the running time is $n^{O(1/\epsilon)}$. This result combines the LP-based 0.25-approximation algorithm (mentioned above) with an enumeration method that ensures each vertex/product has a small weight of outgoing edges. Because the overall algorithm here is not LP based, we cannot use the approximation framework from Zhang et al. (2020). The second application is based on the observation that the directed cut function is submodular. This allows us to directly use results from Buchbinder and Feldman (2019) on maximizing

nonnegative submodular functions over various constraints (these algorithms are also not LP based). In particular, we can obtain a 0.385-approximation algorithm for a multidimensional knapsack constraint (with constant number of dimensions) and matroid constraints. We note that even for max-dicut under a single knapsack constraint, the best approximation ratio prior to our work was 0.385, which followed from Buchbinder and Feldman (2019).

Fourth, we obtain a randomized polynomial time approximation scheme (PTAS) for the unconstrained assortment optimization problem. For any $\epsilon > 0$, our algorithm provides a $(1-\epsilon)$ approximate solution with high probability in $n^{O(1/\epsilon^2)}$ time. This is based on our binary search framework and a new algorithm for max-dicut instances arising in our "reduction" from the PCL assortment problem. The new max-dicut algorithm uses random sampling and enumeration to formulate a new linear program, which is rounded randomly. Some of the ideas in this algorithm are based on the PTAS for unweighted max-cut on "dense" graphs obtained by Arora et al. (1999). However, the analysis of our algorithm relies crucially on various properties of the max-dicut instances that arise in context of the PCL assortment problem. Assuming the "unique games conjecture" of Khot (2002), one cannot obtain an approximation ratio better than ≈ 0.878 for generic instances of max-dicut. So this result separates the approximability of PCL assortment optimization from max-dicut. We note that unconstrained PCL assortment optimization was shown to be strongly NP-hard by Zhang et al. (2020), so we cannot expect a fully polynomial time approximation scheme for this problem. We also obtain a randomized PTAS for the cardinality-constrained problem when the cardinality is not too small, namely, $c = \Omega(n)$.

Finally, we perform extensive computational experiments on previously used test instances and observe that our algorithms perform very well. For all of our experiments, we compute an upper bound for the optimal value using an LP (see Section 7 for more details) and compare our algorithms against this upper bound. For both the unconstrained and cardinality-constrained versions, our empirical performance is about 99% on average. For the knapsack-constrained version, our average performance is above 96%. We also note that the computational times for all three algorithms remain nearly the same. For the partition constrained assortment problem, we implemented our new binary-search-based approximation framework combined with a local-search algorithm for max-dicut under partition constraints. The average performance is above 99% (again, relative to an LP-based upper bound). We also test our PTAS for the unconstrained version and find that it is practical for moderate-size instances (up to 50 products).

Table 1 lists the best approximation ratios under various constraints for max-dicut and PCL assortment optimization. For each result, we cite the relevant paper or section in this paper.

1.2. Related Work

Operational problems under the PCL model have received a fair amount of recent attention. Li and Webster (2017) studied pricing problems, and Zhang et al. (2020) and Feldman (2017) studied assortment optimization problems. Zhang et al. (2020) proved that even the unconstrained PCL assortment problem is NP-hard and obtained approximation guarantees for the unconstrained and cardinality-constrained versions (0.79 and 0.25, respectively). Feldman (2017) obtained a 0.083 approximation guarantee for the knapsack-constrained PCL assortment problem.

There is a considerable amount of work in the transportation literature that uses the PCL model. Koppleman and Wen (2000) conducted empirical analysis on travel-mode data and observed that the PCL model achieved higher log-likelihood than MNL and NL models, which indicates that the PCL model fits the data better. Chen et al. (2014) and Karoonsoontawong and Lin (2015) considered traffic equilibrium problems under the PCL model and discussed the benefits of using this model. This research shows that the PCL model outperforms the MNL and NL models in predicting user behavior, especially when there is a complex correlation structure amongst the alternatives.

There has been a lot of work on assortment optimization and pricing problems under the multinomial logit and nested logit models. Exact polynomial-time algorithms for the unconstrained and cardinality-constrained problems under the MNL model were obtained by Talluri and van Ryzin (2004) and Rusmevichientong et al. (2010). Davis et al. (2013) used a linear program formulation to solve the MNL assortment problem under "totally unimodular" constraints.

Davis et al. (2014) studied the assortment optimization problem under the NL model. They obtained polynomial-time algorithms in some special cases and proved the problem to be NP-hard in general. Gallego and Topaloglu (2014) and Feldman and Topaloglu (2015) studied the cardinality- and knapsack-constrained versions of the NL assortment optimization problem.

Assortment optimization has also been studied under other models, such as mixtures of MNL and the Markov chain choice model. Rusmevichientong et al. (2014) showed that the assortment optimization problem under a mixture of MNL models is *NP*-hard and

Table 1. Summary of Results: Max-Dicut vs. PCL Assortment Optimization

Constraint type	Max-dicut	PCL-AO
Unconstrained	0.874 (Lewin et al. 2002)	$\mathbf{1-\epsilon}$ (Section 6)
Cardinality	0.5 (Ageev et al. 2001)	**0.5** (Section 4.4)
		$\mathbf{1-\epsilon}$ if $c=\Omega(n)$ (Section 6)
Knapsack	$\mathbf{0.5-\epsilon}$ (Section 5.1)	$\mathbf{0.5-\epsilon}$ (Section 5.1)
Partition/multidimensional knapsack	0.385 (Buchbinder and Feldman 2019)	$\mathbf{0.385-\epsilon}$ (Section 5.2)

Note. Our results are in bold.

provided approximation algorithms. Blanchet et al. (2016) obtained an efficient algorithm for assortment optimization under the Markov chain choice model.

The max-dicut problem has been widely studied in optimization and theoretical computer science. If the underlying graph is undirected, we obtain the maximum cut (max-cut) problem, which is a special case of max-dicut. The seminal work of Goemans and Williamson (1995) used semidefinite programming to obtain approximation ratios of 0.878 and 0.796 for the max-cut and max-dicut problems, respectively. The guarantee for max-dicut was later improved to 0.874 by Lewin et al. (2002). Max-dicut is NP-hard, and Håstad (2001) showed that one cannot obtain an approximation ratio less than ≈ 0.917 assuming P$\neq$ NP. In fact, assuming the unique-games-conjecture of Khot (2002), Khot et al. (2007) showed that one cannot approximate max-cut better than ≈ 0.878. Ageev et al. (2001) studied the max-dicut problem with a cardinality constraint, where one wants to find a vertex subset of a given size. Using structural properties of the max-dicut LP and the pipage rounding technique, they developed a 0.5−approximation algorithm for this problem.

1.3. Organization

In Section 2, we set up the notation and formulate all the versions of the assortment optimization problem. In Section 3, we describe our key reduction from the assortment optimization problem to the max-dicut problem. In Section 4, we first review the requisite results from Zhang et al. (2020) and then present our new algorithms for the unconstrained, cardinality-constrained, and knapsack-constrained versions. In Section 5, we present the new binary-search framework that can handle more complex constraints. We use this framework to obtain a better approximation for the knapsack-constrained problem and the first constant-factor approximation for partition and multidimensional knapsack constraints. In Section 6, we obtain the randomized PTAS for the unconstrained assortment optimization problem. We present computational results in Section 7. All missing proofs appear in the online appendices.

2. Preliminaries

The set of products is denoted by $[n]=\{1,2,\ldots,n\}$. Each product i has a revenue $r_i \geq 0$ and a preference weight $v_i \geq 0$ associated with it. Let v_0 be the preference weight associated with the option of not making a purchase. The set $M=\{(i,j) \mid i \neq j\}$ of ordered pairs denotes the collection of nests. For each nest $(i,j) \in M$, we let $\gamma_{ij} \in [0,1]$ be the dissimilarity parameter of that nest. The dissimilarity parameter γ_{ij} characterizes the correlation between the utilities of products i and j. We will use the vector $\mathbf{x} \in \{0,1\}^n$ to denote the set of products that are offered. The i^{th} entry of $\mathbf{x}$ is one if, and only if, product i is offered.

We model customer behavior using the paired combinatorial logit model. Here, a purchase decision can be viewed as a two-stage process. In the first stage, the customer either picks one of the $n(n-1)$ nests or decides to not buy a product. The decision to not buy a product is referred to as the *no-purchase* option. The preference weight of each nest (i, j) is $V_{ij}(\mathbf{x})^{\gamma_{ij}}$, where $V_{ij}(\mathbf{x})=v_i^{1/\gamma_{ij}}x_i+v_j^{1/\gamma_{ij}}x_j$. Thus, given $\mathbf{x}$, the probability that a customer picks nest (i, j) is

$$\frac{V_{ij}(\mathbf{x})^{\gamma_{ij}}}{v_0+\sum_{(k,l)\in M}V_{kl}(\mathbf{x})^{\gamma_{kl}}}.$$

Then, in the second stage, given that the customer picks nest (i, j), he or she picks either product i or j. Conditioned on the customer picking nest (i, j), the probability that product i is picked is

$$\frac{v_i^{1/\gamma_{ij}}x_i}{v_i^{1/\gamma_{ij}}x_i+v_j^{1/\gamma_{ij}}x_j},$$

and the probability that product j is picked can be calculated analogously. Now, we let $R_{ij}(\mathbf{x})$ be the expected revenue obtained from nest (i, j) given that the customer has chosen nest (i, j); by the law of total expectation, we have

$$R_{ij}(\mathbf{x})=\frac{r_iv_i^{1/\gamma_{ij}}x_i+r_jv_j^{1/\gamma_{ij}}x_j}{v_i^{1/\gamma_{ij}}x_i+v_j^{1/\gamma_{ij}}x_j}=\frac{r_iv_i^{1/\gamma_{ij}}x_i+r_jv_j^{1/\gamma_{ij}}x_j}{V_{ij}(\mathbf{x})}.$$

We will use $\pi(\mathbf{x})$ to denote the expected revenue obtained from a customer when $\mathbf{x}$ is offered. Again, by the law of total expectation, we have

$$\pi(\mathbf{x}) = \sum_{(i,j)\in M} \frac{V_{ij}(\mathbf{x})^{\gamma_{ij}}}{v_0 + \sum_{(k,l)\in M} V_{kl}(\mathbf{x})^{\gamma_{kl}}} R_{ij}(\mathbf{x}) = \frac{\sum_{(i,j)\in M} V_{ij}(\mathbf{x})^{\gamma_{ij}} R_{ij}(\mathbf{x})}{v_0 + \sum_{(k,l)\in M} V_{kl}(\mathbf{x})^{\gamma_{kl}}}. \quad (1)$$

The PCL model is usually defined using *unordered* pairs $M' = \{(i,j) \mid 1 \le i < j \le n\}$ as the nests. In this paper, we use a definition with ordered pairs because it simplifies notation. We note that this is equivalent to the PCL model defined using unordered pairs. Indeed, given an instance of the usual PCL model (where γ_{ij} is defined for unordered pairs), we set $\gamma_{ij} = \gamma_{ji}$ for all $i,j \in [n]$ and scale the preference weight of the no-purchase option by two (i.e., set $v_0' = 2v_0$). This is an equivalent model that has ordered pairs.

Our goal is to find a subset of products, $\mathbf{x}^*$, to offer in order to maximize the expected revenue. Thus our optimization problem is

$$z^* = \pi(\mathbf{x}^*) = \max_{\mathbf{x}\in\mathcal{F}} \pi(\mathbf{x}), \quad (2)$$

where $\mathcal{F} \subseteq \{0,1\}^n$ is the set of all feasible subsets of products. Throughout this paper, we assume that $\mathcal{F}$ is *downward closed*, that is, for any $A \in \mathcal{F}$ and $B \subseteq A$, we also have $B \in \mathcal{F}$. We consider the following specific constraints:

- *Unconstrained*: $\mathcal{F} = \{0,1\}^n$.
- *Cardinality constrained*: $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_i x_i \le c\}$, where c is the limit on the number of products.
- *Knapsack constrained*: $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_i s_i x_i \le c\}$, where each s_i represents the size of product i and c is the limit on total size.
- *Multidimensional knapsack constraint*: $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_{i=1}^n s_{ki} x_i \le c_k \; k = 1,2,\ldots,q\}$, which involves q different knapsack constraints. Here we assume that q is a constant.
- *Partition*: $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_{i\in I_k} x_i \le c_k, \; k = 1,2,\ldots,q\}$, where the sets $\{I_k\}_{k=1}^q$ form a partition of the products $[n]$, c_k is the limit on the number of products from part k, and q is the total number of parts. Here, the number q of parts can be arbitrarily large.

3. Relating Assortment Optimization to Max-Dicut

We first define functions $h_z : \{0,1\}^n \to \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$ as

$$h_z(\mathbf{x}) = \sum_{(i,j)\in M} V_{ij}(\mathbf{x})^{\gamma_{ij}}(R_{ij}(\mathbf{x}) - z) \text{ for } \mathbf{x} \in \{0,1\}^n \quad \text{and} \quad f(z) = \max_{\mathbf{x}\in\mathcal{F}} h_z(\mathbf{x}) \text{ for } z \in \mathbb{R}. \quad (3)$$

The reason to consider these functions is because the original objective $\pi(\mathbf{x})$ is a ratio (see (1)), which can be "linearized" as follows. For any $z \in \mathbb{R}$ and $x \in \mathcal{F}$, we have $\pi(\mathbf{x}) \ge z$ if and only if $h_z(\mathbf{x}) \ge v_0 \cdot z$. So the optimal value z^* (see (2)) is the maximal value of z such that $\max_{\mathbf{x}\in\mathcal{F}} \pi(\mathbf{x}) \ge z$, which is equivalent to $f(z) \ge v_0 \cdot z$. In this section, we relate the function $f(z)$ to the max-dicut problem, which is defined as follows.

Definition 1 (Maximum Directed Cut). Given a directed graph (V, E) with edge weights $w : E \to \mathbb{R}$, the goal in max-dicut is to select a vertex subset that maximizes the total weight of outgoing edges, that is,

$$\max_{S\subseteq V} \sum_{(i,j)\in E: i\in S, j\notin S} w_{ij}.$$

It is usually assumed that all edge weights are nonnegative.

For any fixed z, we show how one can reduce the problem of finding an optimal $\mathbf{x}$ for $f(z)$ to the max-dicut problem on an appropriately defined graph. The reduction creates a vertex for each product and an additional dummy vertex that will never be selected. There are edges associated with every pair of vertices. The edge weights are defined such that for any subset of products, their objective value in h_z equals the value of the corresponding cut in the graph.

The max-dicut instance is defined on a graph with $n+1$ vertices, where vertices $[n] := \{1,2,\cdots n\}$ represent the products and vertex $n+1$ is a dummy vertex. We associate a binary decision variable x_i with each of the vertices $i \in \{1,2,\cdots n+1\}$, which represents whether or not each vertex is included in the cut. To ensure that the dummy vertex is never included in the cut, we set $x_{n+1} = 0$. In order to reduce notation, let $\overline{r_i} = r_i - z$ for all $i \in [n]$.

For every *ordered* pair (i, j) of products, we introduce four edges: (i, j), (j, i), $(i, n+1)$, and $(j, n+1)$. The weights of these edges are chosen so that they correspond to the contribution of the (i, j) term in function h_z; this is formalized in Lemma 1. In particular, let

$$\xi_{ij}^+ = \frac{\overline{r_i} v_i^{1/\gamma_{ij}}}{(v_i^{1/\gamma_{ij}} + v_j^{1/\gamma_{ij}})^{1-\gamma_{ij}}} \quad \text{and} \quad \xi_{ji}^- = \frac{\overline{r_j} v_j^{1/\gamma_{ij}}}{(v_i^{1/\gamma_{ij}} + v_j^{1/\gamma_{ij}})^{1-\gamma_{ij}}}. \quad (4)$$

We set ξ_{ij}^+ and ξ_{ji}^- to be the weights on the edges $(i, n+1)$ and $(j, n+1)$, respectively, corresponding to the ordered pair (i, j). Likewise, ξ_{ij}^- and ξ_{ji}^+ are the weights on edges $(i, n+1)$ and $(j, n+1)$, respectively, corresponding to the ordered pair (j, i).

Further, let

$$\psi_{ij}^+ = v_i\overline{r_i} - \xi_{ij}^+ = v_i\overline{r_i}\left(1 - \frac{v_i^{1/\gamma_{ij}-1}}{(v_i^{1/\gamma_{ij}} + v_j^{1/\gamma_{ij}})^{1-\gamma_{ij}}}\right) \text{ and}$$

$$\psi_{ij}^- = v_j\overline{r_j} - \xi_{ji}^- = v_j\overline{r_j}\left(1 - \frac{v_j^{1/\gamma_{ij}-1}}{(v_i^{1/\gamma_{ij}} + v_j^{1/\gamma_{ij}})^{1-\gamma_{ij}}}\right). \quad (5)$$

We set ψ_{ij}^+ and ψ_{ji}^- to be the weights on edges (i, j) and (j, i), respectively (corresponding to the pair (i, j)).

Likewise, ψ_{ij}^- and ψ_{ji}^+ are the weights on edges (i, j) and (j, i), respectively, because of the pair (j, i). Figure 1(a) represents this construction for a particular pair (i, j). Note that every edge e is assigned weight because of multiple (i, j) pairs. We define the final weight of an edge e as the sum of its weights because of all ordered (i, j) pairs. Formally, the final weight of each edge (i, j) is

$$w_{ij}(z) = \begin{cases} \sum_{k\in[n]\setminus\{i\}} \xi_{ik}, & \text{if } i \in [n],\ j = n+1 \\ \psi_{ij}, & \text{if } i,j \in [n], \end{cases} \tag{6}$$

where $\xi_{ij} = \xi_{ij}^+ + \xi_{ij}^-$ and $\psi_{ij} = \psi_{ij}^+ + \psi_{ij}^-$. Note that these are all linear functions of z. Figure 1(b) shows a partial structure of the directed graph that is obtained after the reduction.

Let G_z be the weighted graph obtained in the above reduction. The vertices are $[n+1] := \{1, 2, \cdots, n, n+1\}$ (also denoted by V) and the edge-set is denoted E.

Lemma 1. *For any subset $S \subseteq [n]$, we have $h_z(\mathbf{1}_S) = \sum_{(i,j)\in E: i\in S, j\notin S} w_{ij}(z)$, where $\mathbf{1}_S \in \{0,1\}^n$ is the indicator vector for subset S.*

We note that graph G_z may have negative weights, but there is additional structure.

Lemma 2. *If $\overline{r}_i < 0$ (respectively $\overline{r}_i \geq 0$) for a vertex i, then every edge leaving vertex i in G_z has a negative (resp. nonnegative) weight.*

For ease of notation, we define the following two sets: $N_z := \{i \in [n] \mid \overline{r}_i < 0\}$ and $P_z := V \setminus N_z$. Note that N_z consists of those vertices that have negative weight outgoing edges. The dummy vertex has no edges leaving it and thus is in the set P_z. Let subgraph $\widetilde{G}_z$ consist of all nonnegative weight edges in G_z. Note that $\widetilde{G}_z$ contains all edges leaving vertices of P_z. Next, we show an equivalence between $f(z)$ and the maximum directed cut problem on G_z.

Consider the following two formulations:

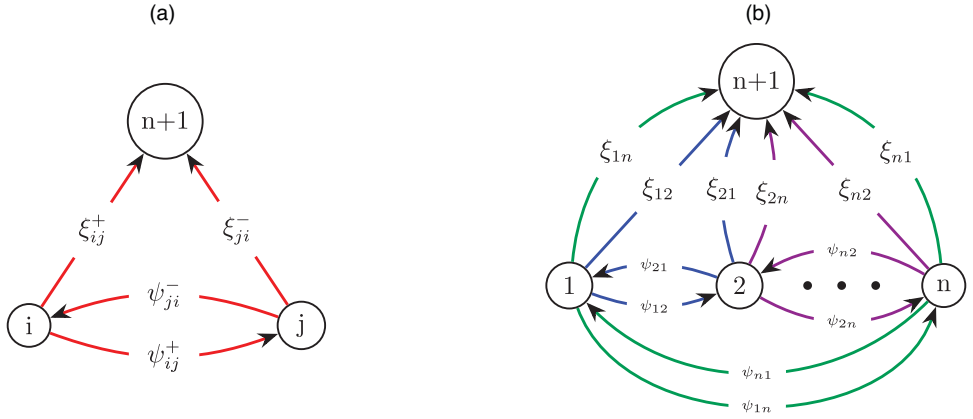
$$\begin{aligned} \text{maximize}: & \sum_{(i,j)\in M} \frac{\overline{r}_i v_i^{1/\gamma_{ij}} x_i + \overline{r}_j v_j^{1/\gamma_{ij}} x_j}{V_{ij}(\mathbf{x})^{1-\gamma_{ij}}} \\ \text{subject to}: & \ \mathbf{x} \in \{0,1\}^n \qquad (\text{IP}_1(z)) \\ & \ \mathbf{x} \in \mathcal{F}, \end{aligned}$$

$$\begin{aligned} \text{maximize}: & \sum_{(i,j)\in E} w_{ij}(z) \cdot y_{ij} \\ \text{subject to}: & \ y_{ij} \leq x_i, && \forall (i,j) \in E \\ & \ y_{ij} \leq 1 - x_j, && \forall (i,j) \in E \\ & \ y_{ij} \in \{0,1\}, && \forall (i,j) \in E \qquad (\text{IP}_2(z)) \\ & \ x_i \in \{0,1\}, && \forall i \in V \\ & \ x_{n+1} = 0 \\ & \ \mathbf{x} \in \mathcal{F}, \end{aligned}$$

where $\mathcal{F}$ represents the set of feasible solutions to the assortment problem. Note that a feasible solution for $\text{IP}_2(z)$ is $(\mathbf{x}, x_{n+1}) \in \mathcal{F} \times \{0\}$. We now argue that there are optimal solutions for $\text{IP}_1(z)$ and $\text{IP}_2(z)$ such that no $i \in N_z$ is included in the optimal assortment or in the optimal cut.

Lemma 3. *Given an optimal solution $\mathbf{x}^*$ to $\text{IP}_1(z)$, we can construct another optimal solution $\overline{\mathbf{x}}$ such that for all $i \in N_z$, $\overline{x}_i = 0$, that is, no product with $\overline{r}_i < 0$ contributes to the solution, and for all $i \in P_z$ (excluding $n+1$, because there are only n products), $\overline{x}_i = x_i^*$.*

Figure 1. (Color online) Structure of the Graph G_z



Notes. (a) Edges added for pair (i, j). (b) Overview of all edges.

Lemma 4. *Given an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to $IP_2(z)$, we can construct another optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ such that for all $i \in N_z$, $\overline{x}_i = 0$, that is, no vertex that has outgoing edges of negative weight contributes to the solution, and for all $i \in P_z$, $\overline{x}_i = x_i^*$.*

Lemma 5. *The optimal value of $IP_1(z)$ is exactly $f(z)$. The optimal value of $IP_2(z)$ is the optimal max-dicut value on G_z subject to the constraints that the dummy vertex is not selected and that the set of chosen vertices lies in $\mathcal{F}$.*

Note also that both $IP_1(z)$ and $IP_2(z)$ are functions of z.

Lemma 6. *The formulations $IP_1(z)$ and $IP_2(z)$ are equivalent, that is, for all $z \in \mathbb{R}$, the optimal value of $IP_1(z)$ is the same as the optimal value of $IP_2(z)$. Moreover, for each $z \in \mathbb{R}$, the optimal value of $IP_2(z)$ equals the optimal max-dicut value on $\widetilde{G_z}$ subject to the constraints that the dummy vertex is not selected and that the set of chosen vertices lies in $\mathcal{F}$.*

3.1. Complexity of the Reduction

We note that even when the input to the assortment problem (values r_i, v_i, and γ_{ij}) is rational, the reduction above may not be polynomial sized in the input, because the weights $w_{ij}(z)$ may be irrational. As we saw in the proof of Lemma 2, we can write $w_{ij}(z) = \overline{r_i} \cdot \alpha_{ij}$ where $\alpha_{ij} \geq 0$. In Online Appendix A.1, we give an approach to deal with irrational α_{ij} values using Dirichlet's approximation theorem. The idea is to compute rational values $\overline{\alpha_{ij}}$ with polynomial bit complexity such that $|\alpha_{ij} - \overline{\alpha_{ij}}| \leq \epsilon$ for $\epsilon = \frac{1}{r_{\max} n^3}$. Then, we use $\overline{w_{ij}}(z) = \overline{\alpha_{ij}}(r_i - z)$ in all our algorithms (instead of $w_{ij}(z)$). As a consequence of this, at the loss of a $1 + o(1)$ factor in the objective, we can assume that all edge weights are rational in the reduction to max-dicut.

Based on the above relation to max-dicut, we obtain the following result (proved in Section 5).

Theorem 1. *Let $\alpha_{\mathcal{C}}$ be the approximation guarantee known for the max-dicut problem under some constraint $\mathcal{C}$. Then, there is an $(\alpha_{\mathcal{C}} - \delta)$–approximation guarantee (for any $\delta > 0$) for the assortment optimization problem under constraints $\mathcal{C}$ in time $\mathcal{O}(\log(\rho_r \rho_v / \delta) \cdot p(n))$ where $p(n)$ is the runtime of the constrained max-dicut algorithm, $\rho_r = \frac{\max_i(r_i)}{\min_i(r_i)}$ and $\rho_v = \frac{\max_i(v_i)}{\min_i(v_i)}$.*

4. Unconstrained, Cardinality, and Knapsack Constraints

In this section, we utilize the approximation framework from Zhang et al. (2020) along with our reduction to max-dicut and new rounding algorithms to design approximation algorithms for the assortment optimization problem. The approximation framework has the following three steps:

1. Construct an upper bound $g(z)$ on $f(z)$ such that $g(z) \geq f(z)$ for all $z \in \mathbb{R}$.
2. Find the fixed point $\hat{z}$ of $g(\cdot)/v_0$; that is, find a $\hat{z} \geq 0$ such that $g(\hat{z}) = v_0 \hat{z}$.
3. Find an assortment $\hat{\mathbf{x}} \in \mathcal{F}$ such that $\sum_{(i,j) \in M} V_{ij}(\hat{\mathbf{x}})^{\gamma_{ij}} (R_{ij}(\hat{\mathbf{x}}) - \hat{z}) \geq \alpha g(\hat{z})$ for some $\alpha \in [0,1]$.

Theorem 2 (theorem 3.1 from Zhang et al. 2020). *Assume that we have an algorithm that satisfies steps 1–3 mentioned above. Then, we have $\pi(\hat{\mathbf{x}}) \geq \alpha \hat{z} \geq \alpha z^*$, where z^* is the optimal value. Thus, $\hat{z}$ is an upper bound to the optimal value of the assortment optimization problem and $\hat{\mathbf{x}}$ is an α–approximate solution to the assortment optimization problem.*

To use Theorem 2, we first find the fixed point of an LP or SDP relaxation (an upper bound to $f(z)$). To this end, as in Zhang et al. (2020), we use the dual LP/SDP to compute the fixed point of the upper bound. We emphasize that the LP/SDP relaxations used in this paper are based on the relation to max-dicut and are different from those used earlier. Finally, to implement step 3 in this approximation framework, we need to round the LP/SDP solutions: this is the key step that also determines the approximation ratio α. Our main contribution is in this step.

Compared with the approximation framework in Theorem 1 (which is based on binary search), the above framework is better if there are LP/SDP-based approximation algorithms for max-dicut. This is because the framework in Theorem 2 only requires solving two LP/SDPs as opposed to a logarithmic number in Theorem 1. This is indeed the case for the algorithms studied in this section. On the other hand, Theorem 2 is inapplicable if the max-dicut approximation algorithms are not LP/SDP based (as for the applications in Sections 5 and 6), whereas the Theorem 1 is still useful.

For the rest of this section, the set $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_i s_i x_i \leq c\}$, which simultaneously captures the unconstrained, cardinality-constrained, and knapsack-constrained versions.

4.1. Relating $f(z)$ to Max-Dicut

We now relax $IP_2(z)$ into a linear program by letting variables take values in $[0,1]$ and by explicitly stating the constraint for $\mathbf{x} \in \mathcal{F}$.

$$
\begin{aligned}
g(z) = \text{maximize}: \quad & \sum_{(i,j) \in E} y_{ij} w_{ij}(z) & & \\
\text{subject to}: \quad & y_{ij} \leq x_i, & \forall (i,j) \in E & \\
& y_{ij} \leq 1 - x_j, & \forall (i,j) \in E & \\
& x_i \leq 1, & \forall i \in [n] & \qquad (\text{LP}(z)) \\
& \sum_{i \in [n]} s_i x_i \leq c & & \\
& x_{n+1} = 0 & & \\
& \mathbf{x}, \mathbf{y} \geq 0. & &
\end{aligned}
$$

Clearly, $g(z)$ is an upper bound to $f(z)$ for all $z \in \mathbb{R}$, as required in step 1 of Theorem 2. Recall that the graph G_z may have negative weight edges. The following lemma provides an algorithm to get a solution for LP(z) where no negative weight edge contributes to the value of the cut. Such a solution can then be used in the rounding procedures described in the future sections.

Lemma 7. *Given an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to LP(z), we can construct another optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ such that for all $i \in N_z$, $\overline{x}_i = 0$, that is, no vertex that has outgoing edges of negative weight contributes to the solution, and for all $i \in P_z$, $\overline{x}_i = x_i^*$.*

The proof of Lemma 7 is identical to the proof of Lemma 4 and hence omitted. As a consequence of Lemma 7, we can, without loss of generality, delete all negative weight edges in G_z to obtain graph $\widetilde{G}_z$. Note that $\widetilde{G}_z$ has nonnegative edge weights and, hence, will be a valid input for the approximation algorithms used later. Moreover, because of Lemma 6, a maximum directed cut in $\widetilde{G}_z$ corresponds to an assortment that maximizes $f(z)$.

We would like to point out that the edge weights of the graph have a negative linear dependence on the parameter z, and thus the following observation is immediate.

Observation 1. Function $g(z)$ is nonincreasing in z.

Lemma 8. *There exists a fixed point for $g(\cdot)/v_0$.*

The rest of this section is organized as follows. We describe the overall algorithm in Section 4.2. We then provide some preliminary results on the LP-rounding approach that we use in Section 4.3. Finally, Section 4.4 addresses the cardinality-constrained and knapsack-constrained problems.

4.2. The Overall Approach

Continuing with the approach from Theorem 2, we now compute a fixed point for the upper bound that we have constructed for $f(z)$. We first write the dual of LP(z) using variables α_{ij}, β_{ij}, γ_i, and δ for the constraints in LP(z).

$$\begin{aligned}
\text{minimize:}\quad & \sum_{(i,j)\in E} \beta_{i,j} + \sum_{i\in[n]} \gamma_i + c\delta \\
\text{subject to:}\quad & \alpha_{ij} + \beta_{ij} \geq w_{ij}(z), \qquad \forall\, (i,j) \in E \\
& -\sum_{j|(i,j)\in E} \alpha_{ij} + \sum_{j|(j,i)\in E} \beta_{ji} + \gamma_i + s_i\delta \geq 0, \\
& \qquad\qquad \forall\, i \in [n] \\
& \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \delta \geq 0 \qquad\qquad \text{(D(z))}
\end{aligned}$$

In the above dual, z is fixed and can be thought of as an input parameter. Next, we add a constraint to the above dual and make z a decision variable. Consider the following LP, which is obtained from the above dual:

$$\begin{aligned}
\text{minimize:}\quad & \sum_{(i,j)\in E} \beta_{i,j} + \sum_{i\in[n]} \gamma_i + c\delta \\
\text{subject to:}\quad & \alpha_{ij} + \beta_{ij} \geq w_{ij}(z), \qquad \forall\, (i,j) \in E \\
& -\sum_{j|(i,j)\in E} \alpha_{ij} + \sum_{j|(j,i)\in E} \beta_{ji} + \gamma_i + s_i\delta \geq 0, \\
& \qquad\qquad \forall\, i \in [n] \\
& \sum_{(i,j)\in E} \beta_{i,j} + \sum_{i\in[n]} \gamma_i + c\delta = v_0 z \\
& \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \delta \geq 0, z \text{ free.} \qquad\qquad \text{(D)}
\end{aligned}$$

Lemma 9. *If $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}, \hat{\delta}, \hat{z})$ is an optimal solution to D, $g(\hat{z}) = v_0\hat{z}$, that is, $\hat{z}$ is the fixed point of $g(\cdot)/v_0$.*

We now summarize the overall algorithm for knapsack-constrained assortment optimization.

1. Solve the LP (D).
2. Using the value of z obtained in the above solution, solve (LP(z)) to obtain $(\mathbf{x}^*, \mathbf{y}^*)$.
3. Using Lemma 7, find another optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ to (LP(z)), where $\overline{x}_i = 0$ for all $i \in N_z$.
4. Round the (LP(z)) solution $\overline{\mathbf{x}}$ to the constrained max-dicut instance on graph $\widetilde{G}_z$ with *nonnegative* edge-weights.

Therefore, if we have an LP-based α-approximation algorithm for constrained max-dicut with nonnegative edge-weights, then we obtain an α-approximation algorithm for the constrained PCL assortment optimization problem. Moreover, this approach only requires us to solve two LPs.

4.2.1. Application to the Unconstrained Problem. For the unconstrained version, we use a semidefinite program relaxation as the upper-bound $g(z)$. The overall algorithm remains the same as above, except for the use of SDPs in steps 1 and 2 and an SDP-based lemma (similar to Lemma 7) that ensures that we can focus on nonnegative edge-weights. We note that Zhang et al. (2020) also used an SDP relaxation to obtain their 0.79-approximation algorithm. However, our SDP and its rounding are different. We obtain the following result (proved in Online Appendix B).

Theorem 3. *There is a randomized 0.874-approximation algorithm for the unconstrained PCL assortment optimization problem.*

4.3. Preliminary Results for the LP Rounding Algorithms

Here we introduce a useful LP rounding technique called pipage rounding that will be used in the algorithms for cardinality-constrained and knapsack-

constrained assortment problems. This technique was developed for problems in which we want the feasible set to be of a given cardinality. The interested reader can refer to Ageev and Sviridenko (1999) for more details. Here, we consider a knapsack constraint on the feasible sets, which involves arbitrary sizes.

Let a problem Π be represented as the program (7).

$$\begin{aligned} &\text{maximize:} && F(\mathbf{x}) \\ &\text{subject to:} && x_i \in \{0,1\}, \quad \forall\, i \in V \\ &&& \sum_i s_i x_i \le c \end{aligned} \tag{7}$$

$$\begin{aligned} &\text{maximize:} && L(\mathbf{x}) \\ &\text{subject to:} && x_i \in [0,1], \quad \forall\, i \in V \\ &&& \sum_i s_i x_i \le c, \end{aligned} \tag{8}$$

where $F : [0,1]^n \to \mathbb{R}_+$ is a continuous function on the n-dimensional cube and L is a linear function. We call LP (8) a "nice" relaxation to (7) if $L(\mathbf{x}) = F(\mathbf{x})$ for all $\mathbf{x} \in \{0,1\}^n$. For solution $\mathbf{x} \in [0,1]^n$, we say that the knapsack constraint is *active* if it is satisfied at equality, that is, $\sum_i s_i x_i = c$.

We define two properties:

Definition 2. F/L lower bound: For a given $\mathbf{x} \in [0,1]^n$, $F(\mathbf{x}) \ge \alpha L(\mathbf{x})$.

Definition 3. $\epsilon-$convexity: For each $i,j \in [n]$ and $\mathbf{x} \in [0,1]^n$, the function $\phi(\mathbf{x},i,j,\epsilon) = F(x_1,\ldots,x_i+\epsilon,\ldots, x_j - \frac{s_i}{s_j}\epsilon,\ldots,x_n)$ is convex in ϵ for $\epsilon \in [-\min\{x_i, \frac{s_j}{s_i}(1-x_j)\}, \min\{(1-x_i), \frac{s_j}{s_i}x_j\}]$.

We now describe the pipage rounding procedure. Let $\overline{\mathbf{x}}$ be any solution to LP (8) that satisfies the condition in Definition 2. Starting with $\mathbf{x} = \overline{\mathbf{x}}$, we repeat the following as long as there are *at least two* fractional variables in $\mathbf{x}$:

1. Pick any two fractional variables, say i and j, that is, $0 < x_i, x_j < 1$.
2. By the $\epsilon-$convexity property, we have $\phi(\mathbf{x},i,j,\epsilon) \ge F(\mathbf{x})$ for either $\epsilon = -\min\{x_i, \frac{s_j}{s_i}(1-x_j)\}$ or $\epsilon = \min\{(1-x_i), \frac{s_j}{s_i}x_j\}$. Let ϵ^* denote the value so that $\phi(\mathbf{x},i,j,\epsilon^*) \ge F(\mathbf{x})$.
3. Obtain a new solution, $\widetilde{\mathbf{x}} = (x_1,\ldots,x_i+\epsilon^*,\ldots,x_j - \frac{s_i}{s_j}\epsilon^*,\ldots,x_n)$. Note that $F(\widetilde{\mathbf{x}}) \ge F(\mathbf{x})$ and either $\widetilde{x}_i \in \{0,1\}$ or $\widetilde{x}_j \in \{0,1\}$.
4. Update $\mathbf{x} \leftarrow \widetilde{\mathbf{x}}$.

Note that the number of fractional variables decreases by at least one in each iteration: so we need to perform at most $n-1$ iterations of the above procedure. Also, the total size in the knapsack constraint remains unchanged throughout, that is, $\sum_i s_i x_i = \sum_i s_i \overline{x}_i$. Let $\mathbf{x}^*$ denote the final solution; note that $F(\mathbf{x}^*) \ge F(\overline{\mathbf{x}})$. Clearly, $\mathbf{x}^*$ has at most one fractional variable. Moreover, if $s_i = 1$ for all $i \in [n]$, $c \in \mathbb{Z}_+$ and the knapsack constraint is active, then it is clear that $\mathbf{x}^*$ cannot have exactly one fractional variable: so $\mathbf{x}^* \in \{0,1\}^n$ in this case. Finally, by the F/L lower bound property for $\overline{\mathbf{x}}$ (i.e., $F(\overline{\mathbf{x}}) \ge \alpha L(\overline{\mathbf{x}})$) and $F(\mathbf{x}^*) \ge F(\overline{\mathbf{x}})$, we have $F(\mathbf{x}^*) \ge \alpha L(\overline{\mathbf{x}})$.

To summarize,

Theorem 4. *Consider an instance of a problem described by (7) with a "nice" relaxation (8) and any solution* $\mathbf{x}$ *to (8). Suppose that F satisfies the* $\epsilon-$*convexity property and the F/L lower bound property for* $\mathbf{x}$ *and some* $\alpha < 1$. *Then we can find a solution* $\mathbf{x}^*$ *with at most one fractional variable such that* $F(\mathbf{x}^*) \ge F(\mathbf{x}) \ge \alpha \cdot L(\mathbf{x})$. *Furthermore, if* $s_i = 1$ *for all* $i \in [n]$, $c \in \mathbb{Z}_+$ *and the knapsack constraint is active for* $\mathbf{x}$, *then* $\mathbf{x}^*$ *is guaranteed to be an integral solution.*

4.3.1. The LP Relaxation for Max-Dicut.

We will apply this method to max-dicut with a knapsack constraint. As the nice relaxation, we use the following linear program:

$$\begin{aligned} &\text{maximize:} && \sum_{(i,j)\in E} w_{ij} \cdot y_{ij} \\ &\text{subject to:} && y_{ij} \le x_i, && \forall\, (i,j) \in E \\ &&& y_{ij} \le 1 - x_j, && \forall\, (i,j) \in E \\ &&& x_i \le 1, && \forall\, i \in V \\ &&& \sum_{i\in[n]} s_i x_i \le c \\ &&& x, y \ge 0. \end{aligned} \tag{9}$$

Note that this LP is equivalent to (LP(z)) when we set $w_{ij} = w_{ij}(z)$. Moreover, as mentioned in Section 4.2, it suffices to consider instances with nonnegative edge-weights.

The following structural result on the basic feasible solutions (x, y) of this LP is crucial in applying the pipage rounding technique. This result is a generalization of a previous result in Ageev et al. (2001) that only holds for cardinality constraints (i.e., every $s_i = 1$).

Theorem 5. *Let* (x, y) *be any basic feasible solution to the LP relaxation (9). Then there is some value* $0 < \delta < \frac{1}{2}$ *such that* $x_i \in \{0, \delta, \frac{1}{2}, 1-\delta, 1\}$ *for all* $i \in V$. *Moreover, if* $\sum_i s_i x_i < c$, *then* $x_i \in \{0, 1/2, 1\}$ *for all* $i \in V$.

4.4. The Cardinality and Knapsack Constrained Problems

We now provide a rounding algorithm for max-dicut with a general knapsack constraint, relative to the LP (9). We formulate this problem as the integer program (7) with $F(\mathbf{x}) = \sum_{(i,j)\in E} w_{ij} x_i (1-x_j)$. We also use $L(\mathbf{x}) = \sum_{(i,j)\in E} w_{ij} \min\{x_i, (1-x_j)\}$ in (8), which is equivalent to LP (9). Note that the $\epsilon-$ convexity property (Definition 3) holds for F because the coefficient on the quadratic term in ϵ is positive. Our algorithm is now given as Algorithm 1.

Lemma 10. *The solution* $\mathbf{x}'$ *in step* 2 *satisfies* $\sum_i s_i x'_i = \sum_i s_i x_i = c$ *and is feasible to LP (9).*

Proof. For any $i \in V_1$, the increase $x'_i - x_i = \min\{1-\delta, (1-\delta)S(V_2)/S(V_1)\} =: \Delta_1$. Similarly, for any $i \in V_2$, we have $x'_i - x_i = -\min\{1-\delta, (1-\delta)S(V_1)/S(V_2)\} =: \Delta_2$. So

$$\begin{aligned}\Delta_1 \cdot S(V_1) &= \min\{(1-\delta)S(V_1), (1-\delta)S(V_2)\} \\ &= -\Delta_2 \cdot S(V_2),\end{aligned}$$

and hence $\sum_{i\in V} s_i(x'_i - x_i) = \Delta_1 \sum_{i\in V_1} s_i + \Delta_2 \sum_{i\in V_2} s_i = 0$. This shows $\sum_i s_i x'_i = \sum_i s_i x_i = c$. Moreover, it is easy to see that $\mathbf{x}' \geq 0$. So $\mathbf{x}'$ is feasible to (9). □

It now follows that the solution $\mathbf{x}^*$ obtained in step 4 by pipage rounding (on either $\mathbf{x}$ or $\mathbf{x}'$) also satisfies the knapsack constraint. The next two lemmas bound the objective value depending on whether or not the knapsack constraint is active. Lemma 11 generalizes a result of Ageev et al. (2001) that focused on the cardinality case (where all $s_i = 1$).

Algorithm 1 (Max-Dicut with a Knapsack Constraint)

1: Let $(\mathbf{x}, z)$ be the optimal basic solution to the LP relaxation (9).
2: If $\sum_i s_i x_i = c$, then we define a new vector $\mathbf{x}'$. Let $V_1 = \{i \mid x_i = \delta\}$, $V_2 = \{i \mid x_i = 1-\delta\}$, $V_3 = \{i \mid x_i = \frac{1}{2}\}$ and $V_4 = \{x_i \mid x_i = 0 \text{ or } 1\}$. Let $S(V_1) = \sum_{i\in V_1} s_i$, and $S(V_2) = \sum_{i\in V_2} s_i$.

$$x'_i = \begin{cases} \min\{1, \delta + (1-\delta)S(V_2)/S(V_1)\}, & \text{if } i \in V_1 \\ \max\{0, (1-\delta) - (1-\delta)S(V_1)/S(V_2)\}, & \text{if } i \in V_2 \\ x_i, & \text{if } i \in V_3 \text{ or } V_4 \end{cases} \tag{10}$$

We set $\hat{\mathbf{x}} \leftarrow \text{argmax}\{F(\mathbf{x}), F(\mathbf{x}')\}$.
3: If $\sum_i s_i x_i < c$, then we set $\hat{\mathbf{x}} \leftarrow \mathbf{x}$.
4: Apply pipage rounding (Theorem 4) to $\hat{\mathbf{x}}$ and obtain solution $\mathbf{x}^*$ with at most one fractional variable (say x^*_i). Let $I_1 = \{j \in V : x^*_j = 1\}$ and $I_2 = \{i\}$.
5: **(Cardinality constrained)** If all $s_i = 1$, then output the better of I_1 and $I_1 \cup I_2$.
6: **(Knapsack constrained)** Otherwise, output the better of I_1 and I_2.

Lemma 11. *If the constraint $\sum_i s_i x_i \leq c$ is active, then $\max\{F(\mathbf{x}), F(\mathbf{x}')\} \geq 0.5 \cdot L(\mathbf{x})$.*

Lemma 12. *If the constraint $\sum_i s_i x_i \leq c$ is not active, then $F(\mathbf{x}) \geq 0.5 \cdot L(\mathbf{x})$.*

We obtain the following guarantees for cardinality and knapsack constraints.

Theorem 6. *Algorithm 1 gives a 0.5-approximation algorithm for max-dicut with a cardinality constraint relative to the LP (9). Hence, there is a 0.5-approximation algorithm for the cardinality-constrained PCL assortment optimization problem.*

Theorem 7. *Algorithm 1 gives a 0.25−approximation guarantee for the max-dicut problem with a knapsack constraint relative to LP (9). Hence, there is a 0.25-approximation algorithm for the knapsack-constrained PCL assortment optimization problem.*

From the above analysis, we also see that if the total weight of edges leaving every vertex was small, then we would not lose much by dropping the fractional vertex. Formally, we have the following:

Definition 4. We define $\delta(i)$ to be the sum of the weights of the edges leaving vertex $i \in V$, that is,

$$\delta(i) = \sum_{(i,j)\in E} w_{ij}.$$

Corollary 1. *If $\mathbf{x}$ is an optimal solution to LP (9) and $\widetilde{V} = \{j \in V : 0 < x_j < 1\}$, Algorithm 1 obtains a solution of weight at least $0.5 \cdot LP - \max_{i\in\widetilde{V}} \delta(i)$, where LP is the optimal LP value.*

Proof. We will show that solution I_1 achieves this guarantee. Firstly, note that Algorithm 1 does not modify integer-valued variables in $\mathbf{x}$. So the fractional variable i in $\mathbf{x}^*$ (if any) must lie in $\widetilde{V}$. From the proof of Theorem 7 (see Online Appendix C), we know that the cut value for I_1 is

$$W_{10} + W_1 \geq W_{10} + f \cdot W_0 + (1-f)W_1 - W_0 = F(\mathbf{x}^*) - W_0.$$

Note that $w_0 =$ total weight of edges from i to $\{i \in V : x^*_i = 0\}$, and here $W_0 \leq \max_{i\in\widetilde{V}} \delta(i)$. The corollary now follows as $F(\mathbf{x}^*) \geq 0.5 \cdot L(\mathbf{x}) = 0.5 \cdot LP$. □

We will show later (in Section 5) that this can be used to obtain an improved $(0.5-\epsilon)$-approximation algorithm for knapsack-constrained max-dicut and PCL assortment optimization.

5. Assortment Optimization Under General Constraints

In this section, we prove Theorem 1 by providing a binary-search-based approximation framework. This can be used to solve PCL assortment optimization under a much larger class of constraints. It is also useful in obtaining improved approximation ratios for the unconstrained and knapsack-constrained versions. Unlike the framework from Zhang et al. (2020, theorem 2), our approach here does not require approximation algorithms relative to LP/SDP relaxations.

Based on the reduction in Section 3, the function value $f(z)$ corresponds to solving a max-dicut instance

on G_z. Although some edge weights in G_z may be negative, Lemmas 5 and 6 imply that $f(z)$ equals the optimal max-dicut value on G_z subject to the constraints that the dummy vertex is not selected and that the set of chosen vertices lies in $\mathcal{F}$. Recall that $\widetilde{G}_z$ is the subgraph of G_z consisting of nonnegative edges. So the (constrained) max-dicut instances that we need to solve will only contain nonnegative edge weights. Let $\mathcal{C}$ be the constraint on feasible assortments (and on feasible cuts in the max-dicut instance). Let $\alpha_{\mathcal{C}}$ denote the approximation guarantee for max-dicut under constraint $\mathcal{C}$: so we can obtain an $\alpha_{\mathcal{C}}$-approximation to $f(z)$ for any z. We use a binary search technique to obtain a value $\widetilde{z}$ that is within an $\alpha_{\mathcal{C}}$ multiplicative factor (and some additive error) of the optimal z^*. Recall that z^* is the expected revenue of the optimal assortment and the fixed point of $f(\cdot)/v_0$. Let $\widetilde{f}(z)$ be the objective value of the $\alpha_{\mathcal{C}}$-approximation algorithm on max-dicut instance G_z with constraint $\mathcal{C}$. Roughly speaking, the binary search identifies the fixed point of the approximating function $\widetilde{f}(\cdot)/v_0$. Finally, we show how the additive error can be absorbed into the multiplicative approximation guarantee by a simple scaling idea. Algorithm 2 describes this procedure formally. In Online Appendix D, we show that it achieves the guarantee stated in Theorem 1.

Algorithm 2 (Assortment Optimization under General Constraints)

1: let $L \leftarrow R_{min} := \frac{\min_i(r_i)\cdot\min_i(v_i)}{2\cdot\max_i(v_i)}$, $R \leftarrow R_{max} := \max_i(r_i)$ and $\epsilon \leftarrow \delta R_{min}$.
2: **while** $R - L \geq \epsilon$ **do**
3: $z \leftarrow \frac{L+R}{2}$
4: let $\hat{\mathbf{x}}$ be an $\alpha_{\mathcal{C}}$-approximate solution to max-dicut under constraint $\mathcal{C}$ on graph $\widetilde{G}_z$.
5: $ALG_z \leftarrow \sum_{(i,j)\in M} V_{ij}(\hat{\mathbf{x}})^{\gamma_{ij}}[R_{ij}(\hat{\mathbf{x}}) - z]$
6: **if** $ALG_z \geq v_0 z$, **then** set $L \leftarrow z$
7: **else** Set $R \leftarrow z$
8: **return** $\alpha_{\mathcal{C}}$-approximate solution to max-dicut on graph $\widetilde{G}_L$ under constraint $\mathcal{C}$.

5.1. Improved Algorithm for Knapsack Constraints

Here, we revisit the knapsack-constrained assortment problem where the feasible subsets are $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_i s_i x_i \leq c\}$. We obtained a 0.25-approximation algorithm for this problem in Section 4.4. We now obtain a better $(0.5-\epsilon)$-approximation algorithm for max-dicut under knapsack constraints (for any constant $\epsilon > 0$), which combined with Theorem 1 yields the same approximation ratio for the assortment problem. Prior to our work, the best approximation ratio even for max-dicut under knapsack constraints was 0.385, which followed from Buchbinder and Feldman (2019).

Our algorithm combines a partial enumeration method with the LP-based pipage rounding algorithm in Section 4.4. For any instance of knapsack-constrained max-dicut (with nonnegative weights w_{ij}), recall from Definition 4 that $\delta(i)$ denotes the total weight of outgoing edges from vertex $i \in V$. Moreover, by the pipage rounding algorithm (see Corollary 1) we can obtain a solution of weight at least $0.5 \cdot OPT - \max_{i\in V}\delta(i)$ where OPT is the optimal value. So, if we could bound all the $\delta(i)$ s by a small fraction of OPT, we would obtain the desired result. To this end, we perform an enumeration step described below.

We assume for now that our algorithm knows a number U such that $OPT \leq U \leq 2 \cdot OPT$ (we will see later how to remove this assumption). Given any $\epsilon > 0$, set $\epsilon' = \epsilon/2$. Vertex $i \in V$ is said to be *heavy* if $\delta(i) \geq \epsilon' U$. Let $H \subseteq V$ denote the set of heavy vertices. We enumerate all subsets $S \subseteq H$ of heavy vertices with $|S| \leq \lceil \frac{4}{\epsilon'} \rceil$ that are feasible, that is, $\sum_{i\in S} s_i \leq c$. Note that there are at most $n^{1+4/\epsilon'}$ possible subsets S: so this enumeration takes polynomial time for any constant ϵ.

For a particular choice of subset S, let $LP(S)$ denote the linear program (9) along with

$$x_i = \begin{cases} 1 & \text{for all } i \in S \\ 0 & \text{for all } i \in H \backslash S. \end{cases}$$

Basically, this fixes the decisions on all heavy vertices. The algorithm solves $LP(S)$ optimally to obtain solution $\mathbf{x}$ and rounds it using Algorithm 1. The set of fractional variables in $\mathbf{x}$ is $\widetilde{V} = \{j \in V : 0 < x_j < 1\} \subseteq V \backslash H$, which implies $\max_{i\in\widetilde{V}}\delta(i) < \epsilon' U$. So Corollary 1 implies that we obtain a cut of weight at least $0.5 \cdot LP^*(S) - \epsilon' U$, where $LP^*(S)$ denotes the optimal value of $LP(S)$.

It remains to show that there is some choice of S (that we enumerate) for which $LP^*(S) \geq OPT$, which relies on upper bounding the number of heavy vertices in an optimal solution.

Lemma 13. *For any $\epsilon > 0$, if T denotes the set of heavy vertices with respect to ϵ in an optimal solution, then $|T| \leq \lceil \frac{4}{\epsilon} \rceil$.*

Proof. Consider a subset $S \subseteq T$ picked uniformly at random and let $cut(S)$ denote the weight of edges cut by solution S. Note that every subset $S \subseteq T$ is still feasible for the knapsack constraint. The *expected* weight of cut edges is

$$\begin{aligned}
\mathbb{E}[\text{cut}(S)] &= \sum_{(i,j):i\in T, j\in V} \mathbf{P}(i \in S,\ j \notin S) \cdot w_{ij} \\
&= \sum_{(i,j):i\in T, j\in T} \mathbf{P}(i \in S,\ j \notin S) \cdot w_{ij} \\
&\quad + \sum_{(i,j):i\in T, j\notin T} \mathbf{P}(i \in S,\ j \notin S) \cdot w_{ij} \\
&= \frac{1}{4} \sum_{(i,j):i\in T, j\in T} w_{ij} + \frac{1}{2} \sum_{(i,j):i\in T, j\notin T} w_{ij} \\
&\geq \frac{1}{4} \sum_{(i,j):i\in T, j\in V} w_{ij} = \frac{1}{4} \sum_{i\in T} \delta(i) \\
&\geq \frac{1}{4} |T| \cdot \epsilon \cdot \text{OPT},
\end{aligned}$$

where the first equality follows by applying linearity of expectation, the third equality follows by independence, and the last equality follows from the definition of $\delta(\cdot)$. The final inequality follows from the definition of a heavy vertex.

Moreover, $\mathbb{E}[\text{cut}(S)] \le OPT$ as S is always feasible. Combining the above two inequalities, we have $OPT \ge \frac{1}{4}|T| \cdot \epsilon \cdot OPT$; rearranging we get $|T| \le \frac{4}{\epsilon} \le \lceil \frac{4}{\epsilon} \rceil$. □

Lastly, we need to ensure that we can find some value U with $OPT \le U \le 2 \cdot OPT$. This is done by a simple enumeration. Note that $w_{max} = \max_{(i,j)\in E} w_{ij} \le OPT \le \sum_{(i,j)\in E} w_{ij} \le n^2 \cdot w_{max}$. We enumerate all values $U \in \{2^k \cdot w_{max} : k = 0, 1, \cdots \lceil \log_2(n^2) \rceil\}$, run our algorithm for each choice, and return the best solution found. Clearly, one of these choices satisfies the desired condition. Moreover, the number of choices is $O(\log n)$.

Theorem 8. *For any $\epsilon > 0$, the above algorithm has an approximation guarantee of $(0.5 - \epsilon)$ for max-dicut with a knapsack constraint. Hence, there is a $(0.5 - \epsilon)$-approximation algorithm for knapsack-constrained assortment optimization for all constant $\epsilon > 0$.*

We note that the LP-based approximation framework (Theorem 2) is not applicable here because our improved approximation algorithm is not relative to any LP.

5.2. Multidimensional Knapsack and Matroid Constraints

It is well known that the directed cut function on any graph with nonnegative edge weights is nonnegative and submodular (but not monotone). Therefore, we can use any algorithm for submodular maximization under constraint $\mathcal{C}$ in order to solve the max-dicut problem under $\mathcal{C}$, which in turn (using Theorem 1) provides an algorithm for the assortment optimization problem under constraint $\mathcal{C}$. Here we consider *matroid* and *knapsack* constraints. In a matroid constraint, we are given a matroid with ground set being the n products and the goal is to select any independent set of products. Lee et al. (2010) gave a simple local search algorithm for maximizing any nonnegative submodular function under a matroid constraint, with an approximation guarantee of 0.25. The approximation ratio was improved to 0.385 by Buchbinder and Feldman (2019); this algorithm is significantly more complex than the local search algorithm. Combined with the result of Kulik et al. (2013), the latter result also implies a 0.385-approximation algorithm for submodular maximization under any constant number of knapsack constraints. Thus, using Theorem 1 we obtain the following:

Theorem 9. *Algorithm 2 provides a 0.385-approximation algorithm for assortment optimization under the PCL model subject to a matroid or constant number of knapsack constraints.*

We now provide some examples of specific constraints for assortment optimization that can be captured by the above result. Consider an electronic retail store that has a certain amount of space to showcase their offered products. Each displayed product occupies a fixed amount of space, and the total display space is limited.

5.2.1. Partition Constraints. Suppose that the products are partitioned into different categories (televisions, laptops, tablets, etc.) and the store wishes to limit the number of displayed products from each category. We can model this situation by introducing partition constraints. Suppose we have q different product-categories and that I_k is the set of products in category $k \in [q]$; we assume that $I_1, \cdots I_q$ is a partition of all products. Let c_k be the limit on the number of displayed products of each category $k \in [q]$. We can then represent the set of feasible subsets as $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_{i\in I_k} x_i \le c_k, k = 1, 2, \ldots, q\}$, which is a matroid constraint.

5.2.2. Hierarchical Partition Constraints. A further generalization of the above application allows product categories to be *nested* and not just disjoint. Formally, if $A \subseteq [n]$ and $B \subseteq [n]$ denote the products of any two categories, then we must have $A \cap B = \emptyset$, $A \subseteq B$, or $B \subseteq A$. The first case ($A \cap B = \emptyset$) corresponds to disjoint categories (as for partition constraints), whereas the other two cases correspond to nested categories. For example, products in the category "television sets" may be subdivided based on screen size or picture quality. Given such a nested (or hierarchical) partition into categories and a limit on the maximum number of products of each category, the goal is to select a subset of products respecting all of these limits. The resulting collection $\mathcal{F}$ of feasible subsets is again a matroid, and we obtain a 0.385-approximation algorithm from Theorem 9.

5.2.3. Multidimensional Knapsack Constraints. More generally, the store may want to enforce constraints involving overlapping sets of products. For example, in addition to limits by categories (as above), one may have limits on the number or space of all products (from any category) having some feature, such as luxury products or on-sale products. Although this situation does not correspond to a matroid constraint, it can be modeled as a *multidimensional* knapsack constraint, with one dimension for each constraint. For each constraint k (representing some category

or feature), let s_{ki} be the space used by product i if it participates in constraint k and zero otherwise and let c_k be the total space allocated to products in constraint k. Then, the feasible subsets are $\mathcal{F} = \{\mathbf{x} \in \{0,1\}^n \mid \sum_{i=1}^n s_{ki} x_i \le c_k \; k = 1, 2, \ldots, q\}$. As long as the number q of constraints is constant, we obtain a 0.385-approximation algorithm from Theorem 9.

6. PTAS for Unconstrained Assortment Optimization

The main result in this section is

Theorem 10. *There is a randomized polynomial-time approximation scheme for the unconstrained assortment optimization problem under the PCL model.*

We will use the binary-search framework (Theorem 1) combined with additional properties of the max-dicut instances that arise from the assortment problem. In particular, we will show that for any $z \ge 0$, there is a randomized PTAS for the function value $f(z)$ in (3). Recall from Section 3 and Lemmas 5 and 6 that the function value $f(z)$ equals the optimal max-dicut value on $\widetilde{G}_z$ (where the dummy vertex is not selected). Our focus here is the unconstrained case: so $\mathcal{F} = \{0,1\}^n$. As before, $\widetilde{G}_z$ is the subgraph of G_z consisting of nonnegative edges.

In order to obtain a PTAS for max-dicut on $\widetilde{G}_z$, we use the "exhaustive sampling" approach of Arora et al. (1999), which involves sampling a random subset of vertices and enumerating whether or not these vertices appear in an optimal solution. This approach has been used to provide PTASes for *dense* instances of max-dicut (as well as other graph optimization problems). In our setting, however, the max-dicut instances are not necessarily dense: so we cannot directly use prior results. Nevertheless, using the structure of our max-dicut instances, we prove a lower bound on its optimal value (see Lemma 14). Moreover, we show that the additive error incurred because of the sampling steps is only a small fraction of this lower bound (see Lemma 16), which leads to a PTAS. As noted in Section 1.2, there is no PTAS for max-dicut in general (unless P = NP).

In our reduction described in Section 3, recall that for every pair $(i,j) \in [n] \times [n]$, we add four edges: $(i, n+1)$, $(j, n+1)$, (i, j), and (j, i) with weights ξ_{ij}^+, ξ_{ji}^-, ψ_{ij}^+, and ψ_{ji}^-, respectively. From the definitions (4) and (5), we have $\xi_{ij}^+ + \psi_{ij}^+ = v_i \bar{r}_i$ and $\xi_{ji}^- + \psi_{ji}^- = v_j \bar{r}_j$. Also recall the edge weights $w_{ij}(z)$ as defined in (6). In particular, for $i,j \in [n]$, we have $w_{ij}(z) = \psi_{ij}^+ + \psi_{ij}^-$. Let $Q_z = \{i \in [n] : \bar{r}_i \ge 0\}$; recall that $P_z = Q_z \cup \{n+1\}$ and $N_z = [n+1] \backslash P_z$. Let $N = |Q_z|$. For any $i \in Q_z$ and $j \in [n]$, we have $w_{ij}(z) \le 2 v_i \bar{r}_i$ based on the calculations in Lemma 2. For easier notation, for all $i,j \in Q_z$, we define $\sigma_{ij} = \frac{w_{ij}(z)}{v_i \bar{r}_i} \in [0,2]$. Also, for any $i \in Q_z$, define $C_i(z) = \sum_{j \notin Q_z} w_{ij}(z)$. Finally, note that the optimal max-dicut on $\widetilde{G}_z$ only contains vertices from Q_z. So $f(z)$ equals the following:

$$\max_{A \subseteq Q_z} \left(\sum_{i \in A} v_i \bar{r}_i \cdot \sum_{j \in Q_z \backslash A} \sigma_{ij} + \sum_{i \in A} C_i(z) \right). \tag{11}$$

Below, we use OPT to refer to the optimal solution to (11) as well as its value. It will be clear from context which of the two is being referenced. Let $\overline{OPT} = Q_z \backslash OPT$, THAT IS, the set of vertices that are not in the optimal max-dicut solution. For every vertex $i \in Q_z$, define $O_i = \sum_{j \in \overline{OPT}} \sigma_{ij}$; note that $O_i \le 2N$. Consequently, the optimal value can be written as

$$OPT = \sum_{i \in OPT} (v_i \bar{r}_i \cdot O_i + C_i(z)).$$

The PTAS starts by sampling a random subset $S \subseteq Q_z$; then for each partition $(U, \overline{U})$ of S, it does the following and picks the best solution found. (1) Define the term $\alpha_i = \sum_{j \in \overline{U}} \sigma_{ij}$ for each $i \in Q_z$. (2) Solve LP(α) (described below). (3) Randomly round the solution obtained on solving LP(α) to obtain an integral solution. The main idea is that when the partition $(U, \overline{U})$ is such that $U = S \cap OPT$ and $\overline{U} = S \cap \overline{OPT}$, the values $\{\alpha_i\}_{i \in Q_z}$ are unbiased estimators of $\{O_i\}_{i \in Q_z}$. Using this, we write a "sampled" linear program (LP(α) below) that has optimal value roughly equal to OPT. Finally, random rounding of this LP solution yields an integral solution of value $\approx OPT$.

Algorithm 3 (Randomized PTAS for Max-Dicut on $\widetilde{G}_z$)

1: Sample S: for every $i \in Q_z$, add i to S independently with probability $p = \min\left\{\frac{96 \log N}{\epsilon^2 N}, 1\right\}$.
2: If $|S| > 2Np$, then set $\hat{\mathbf{x}} = \mathbf{x}^* = \mathbf{0}$ and return solution $\mathbf{x}^*$.
3: **for** all splits $(U, \overline{U})$: $U \subseteq S$, $\overline{U} = S \backslash U$ **do**
4: Let $\alpha_i := \sum_{j \in \overline{U}} \sigma_{ij}$ for all $i \in Q_z$
5: Solve the following LP and let $\hat{\mathbf{x}}$ be its optimal solution.

$$\begin{aligned} \text{maximize:} \quad & \frac{1}{p} \sum_{i \in Q_z} \alpha_i v_i \bar{r}_i \cdot x_i + \sum_{i \in Q_z} C_i(z) \cdot x_i \\ \text{subject to:} \quad & \sum_{j \in Q_z} \sigma_{ij} \cdot (1 - x_j) \ge \frac{\alpha_i}{p} - N\epsilon, \quad \forall i \in Q_z \\ & x_i \in [0,1], \qquad \forall i \in Q_z \end{aligned} \tag{LP(α))}$$

6: From $\hat{\mathbf{x}}$, get integral solution $\mathbf{x}^*$ as follows: $x_i^* = \begin{cases} 1, \text{ with probability } \hat{x}_i \\ 0, \text{ otherwise} \end{cases}$, for all $i \in Q_z$,
7: Return the best $\mathbf{x}^*$ found over all runs.

Theorem 11. *For any $\epsilon \in (0,1)$, Algorithm 3 runs in $n^{O(1/\epsilon^2)}$ time and finds a solution with the expected objective at least $1-8\epsilon$ times the optimal max-dicut value on $\widetilde{G}_z$.*

We first discuss the running time of Algorithm 3. If the algorithm continues beyond step 2, the sample size $|S| \le 2Np \le \frac{192 \log N}{\epsilon^2}$. So the number of iterations of the for-loop, which is the number of possible splits of S, is at most $2^{|S|} = n^{O(1/\epsilon^2)}$. Moreover, each iteration of the for-loop involves solving an LP with N variables and constraints and simple randomized rounding, all of which takes polynomial time. So the overall runtime is $n^{O(1/\epsilon^2)}$ as claimed in Theorem 11.

We now analyze the performance guarantee. Throughout, we will assume that $N \ge \frac{3}{\epsilon}$: otherwise (11) can be easily solved by enumeration. The analysis involves showing that OPT is a feasible solution to LP(α) with high probability, showing a lower bound for OPT with respect to the fractional solution to LP(α) and finally showing that the expected objective of our rounded solution is within a factor of ϵ from OPT. We start with a key property that uses the structure of the max-dicut instance from Section 3.

Lemma 14. *The optimal value $OPT \ge \frac{N-1}{2} \sum_{i \in Q_z} v_i \overline{r}_i \ge \frac{N}{3} \sum_{i \in Q_z} v_i \overline{r}_i$.*

Proof. Consider a random solution $R \subseteq Q_z$ to (11) where each vertex $i \in Q_z$ is in R with probability $\frac{1}{2}$ independently. Using the definition of $C_i(z)$, the objective value of R is

$$obj(R) = \sum_{i \in R} \left(\sum_{j \in Q_z \setminus R} w_{ij}(z) + \sum_{j \in Q_z} w_{ij}(z) \right).$$

By definition of R, for any $i,j \in Q_z$, we have $\mathbf{P}[i \in R] = \frac{1}{2}$ and $\mathbf{P}[i \in R, j \notin R] = \frac{1}{4}$. So,

$$\begin{aligned}
\mathbb{E}[obj(R)] &= \frac{1}{4} \sum_{i,j \in Q_z} w_{ij}(z) + \frac{1}{2} \sum_{i \in Q_z, k \notin Q_z} w_{ik}(z) \\
&\ge \frac{1}{4} \sum_{i \in Q_z} \left(\sum_{j \in Q_z} w_{ij}(z) + w_{i,n+1}(z) \right) \\
&= \frac{1}{4} \sum_{i \in Q_z} \left(\sum_{j \in Q_z} w_{ij}(z) + \sum_{k \in [n] \setminus \{i\}} (\xi_{ik}^+ + \xi_{ik}^-) \right) \qquad \text{(A)} \\
&\ge \frac{1}{4} \sum_{i,j \in Q_z} \left((\psi_{ij}^+ + \psi_{ij}^-) + (\xi_{ij}^+ + \xi_{ij}^-) \right) \qquad \text{(B)} \\
&= \frac{N-1}{4} \sum_{i \in Q_z} 2 v_i \overline{r}_i. \qquad \text{(C)}
\end{aligned}$$

Above, the equality (A) uses the definition of $w_{i,n+1}(z)$ and the inequality in (B) uses the definition of $w_{i,j}(z)$ and that all the terms are nonnegative. The final equality (C) uses the observation that $\xi_{ij}^+ + \psi_{ij}^+ = v_i \overline{r}_i = \xi_{ij}^- + \psi_{ij}^-$ from (5). The lemma now follows as $OPT \ge \mathbb{E}[obj(R)]$. □

For any sample $S \subseteq Q_z$, define its *canonical split* as $(T_S, \overline{T_S})$ of S, where $T_S = S \cap OPT$ and $\overline{T_S} = S \cap \overline{OPT}$. Note that one of the iterations of the for-loop will consider this split, and the algorithm chooses the best solution over all iterations. In the analysis below, we will only consider the solution resulting from the canonical split of S. Specifically, for this split, random variable $\alpha_i = \sum_{j \in S \cap \overline{OPT}} \sigma_{ij}$ and so $\mathbb{E}_S[\alpha_i] = \sum_{j \in \overline{OPT}} \sigma_{ij} \cdot \mathbf{P}[j \in S] = p \cdot O_i$. We now have the following concentration bound.

Lemma 15. *For each $i \in Q_z$, $\Pr_S[|\alpha_i - \mathbb{E}[\alpha_i]| > \epsilon Np] \le \frac{2}{N^3}$ for all $i \in Q_z$.*

So, we have $\alpha_i \in [p \cdot O_i \pm \epsilon Np]$, for all $i \in Q_z$ with high probability, for the canonical split $(T_S, \overline{T_S})$. Let $\hat{\mathbf{x}}$ be the optimal solution for LP(α) for the split $(T_S, \overline{T_S})$ and $LP(\hat{\mathbf{x}})$ be the corresponding value. Then we have the following.

Lemma 16. *The expected value $\mathbb{E}_S[LP(\hat{\mathbf{x}})] \ge (1 - \frac{3}{N^2})\, OPT - \epsilon N \sum_{i \in Q_z} v_i \overline{r}_i$.*

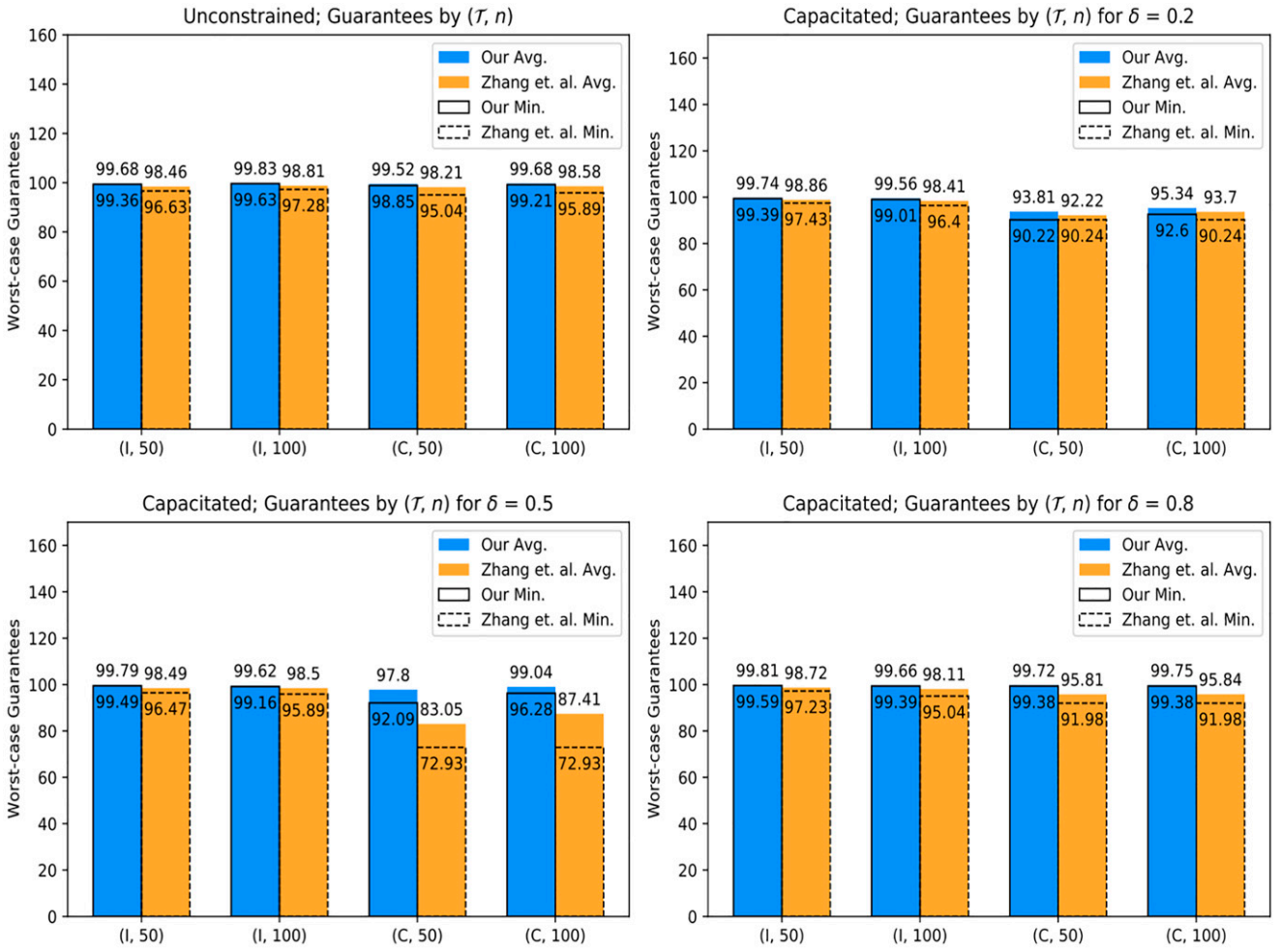
Let ALG denote the objective value in (11) of our algorithm's solution $\mathbf{x}^*$.

Lemma 17. *The expected value $\mathbb{E}_S[ALG] \ge (1 - 8\epsilon) \cdot OPT$.*

This completes the proof of Theorem 11. Note that this provides a guarantee on the expected objective. We can obtain a high-probability guarantee as follows. Let q denote the probability that Algorithm 3 finds a solution of the objective at least $(1-16\epsilon)OPT$. As the algorithm's objective is always at most OPT, its expected objective is at most $q \cdot OPT + (1-q) \cdot (1-16\epsilon)OPT$. From Theorem 11, the expected objective is at least $(1-8\epsilon)OPT$, which implies $q \ge \frac{1}{2}$. By repeating the entire algorithm independently T times and picking the best solution, it follows that we obtain a solution of value at least $(1-16\epsilon)OPT$ with probability at least $1 - 2^{-T}$.

Finally, to prove Theorem 10, we combine this result with the binary-search based framework from Section 5 (see Theorem 1). For any $\epsilon \in (0,1)$, let $\alpha(\epsilon) = 1 - 16\epsilon$ denote the approximation ratio obtained above for the max-dicut instance (11). Setting $\delta = \epsilon$ in Theorem 1, we obtain an $\alpha(\epsilon) - \epsilon = 1 - 17\epsilon$ approximation ratio for the unconstrained PCL assortment problem. This assumes that the randomized PTAS does indeed obtain an $\alpha(\epsilon)$ approximation for every max-dicut instance. Note that the number of max-dicut instances that need to be solved in the algorithm of Theorem 1 is $I = O(\log(\rho_r \rho_v / \epsilon))$. Using $T = O(\log(nI))$ repetitions of Algorithm 3 for each max-dicut instance, it follows that we obtain an $\alpha(\epsilon)$-approximation for *every* instance with probability at least $1 - I \cdot 2^{-T} \ge 1 - \frac{1}{n^2}$. Thus, for any $\epsilon \in (0,1)$, we obtain a $1 - 17\epsilon$ approximation ratio for the PCL assortment

Figure 2. (Color online) Comparing Our Algorithm and Zhang et al. (2020)



Notes. The first plot (top left) is for the unconstrained case. The remaining three plots are for the cardinality-constrained case.

problem with probability at least $1 - o(1)$, which implies Theorem 10.

6.1. Cardinality-Constrained Assortment Optimization

Using the above techniques, and some more ideas, we obtain the following result (see Online Appendix E).

Theorem 12. *There is a randomized polynomial-time approximation scheme for cardinality-constrained assortment optimization in the PCL model when the capacity $c \geq \delta n$, where $\delta > 0$ is some constant.*

7. Computational Results

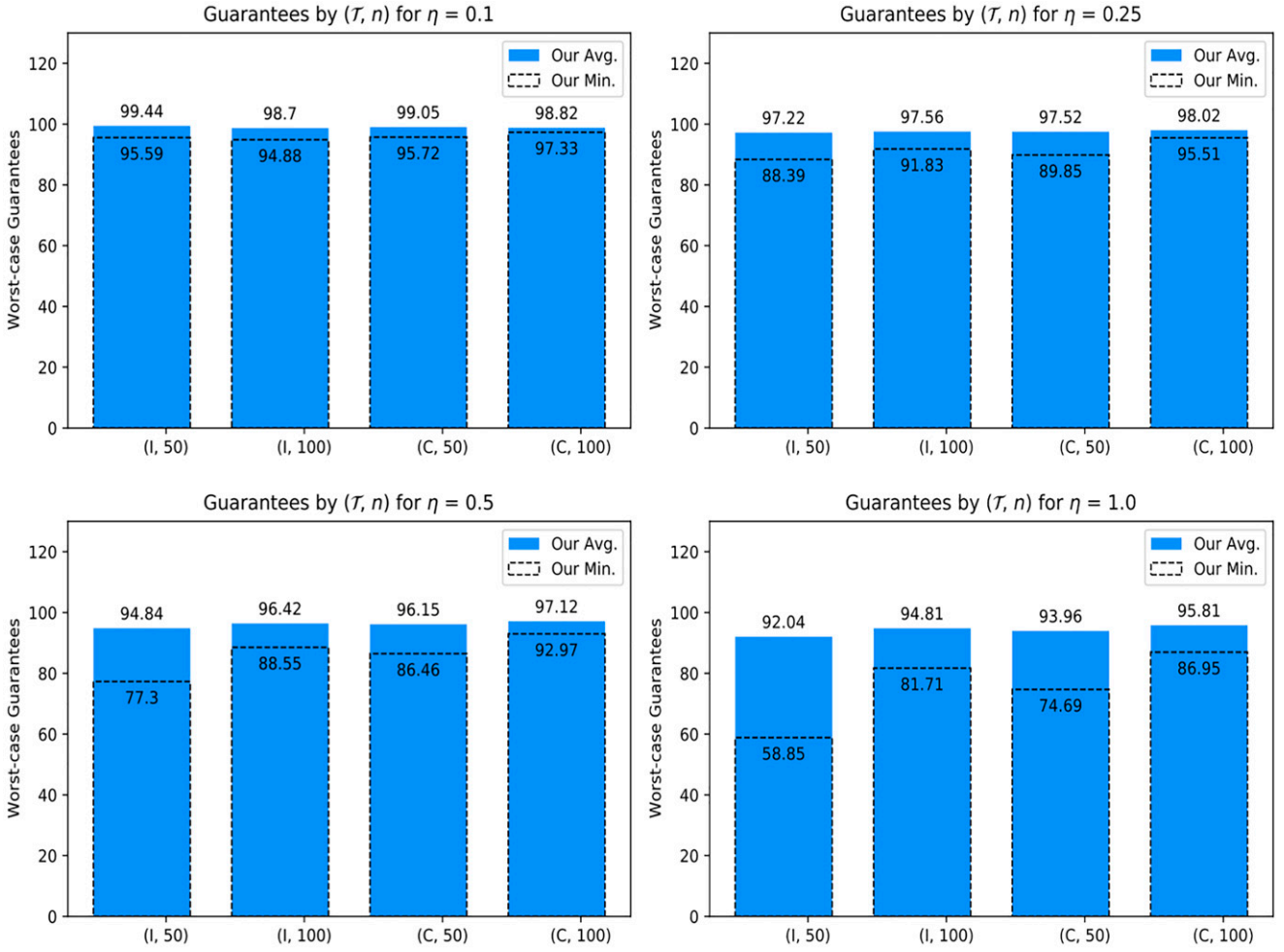
In this section, we provide computational results of our approximation algorithms for unconstrained, cardinality-constrained, knapsack-constrained, and partition-constrained assortment optimization. We conducted all of our computational experiments using Python and Gurobi 8.0 with a 2.3-Ghz Intel Core i5 processor and 16-GB 2133 MHz LPDDR3 memory.

Based on Theorem 2, we use the fixed point $\hat{z}$ of the LP relaxation $g(z)$ as an upper bound. For partition constraints, we replace the knapsack constraint in linear program LP(z) by a set of linear constraints enforcing the partition limits. Although the algorithm for partition constraints does not use such an LP, the value $\hat{z}$ is still a valid upper bound to compare the algorithm against. For each instance, we record our algorithm's performance as $100 \times \pi(\mathbf{x}_{ALG})/\hat{z}$, where $\mathbf{x}_{ALG}$ is our solution. This gives a lower bound (percentage) on how close our solution is to the optimal.

7.1. Instance Generation

We use the same instance-generation procedure as Zhang et al. (2020). The preference weights $\{v_i\}$ for the products are sampled uniformly at random from $[0,1]$. The revenues are sampled in two ways, which lead to two types of problems, independent and correlated. In the independent instances, the revenues $\{r_i\}$ are independently sampled from the uniform distribution on $[0,1]$. In the correlated instances, we set

Figure 3. (Color online) Summary of Results for Knapsack Constrained Assortment Optimization



$r_i = 1 - v_i$, which captures the intuition of more expensive products being less preferable. Parameters I and C refer to independent and correlated instances, respectively. In the unconstrained case, the test problems are labeled as $(T, n, \overline{\gamma}, P_0)$, where T refers to the type of instance, independent or correlated; n denotes the number of products; the dissimilarity parameters are sampled from the uniform distribution $[0, \overline{\gamma}]$; and the value P_0 is used to generate the weight of the no-purchase option. In this way, 36 configurations are generated. For the cardinality-constrained test problems, we have an additional parameter δ; we set the capacity $c = \lceil \delta n \rceil$. These test problems are labeled as $(T, n, \overline{\gamma}, P_0, \delta)$. We generate 72 configurations this way. For each configuration, 100 test instances are generated and solved using our approximation algorithm.

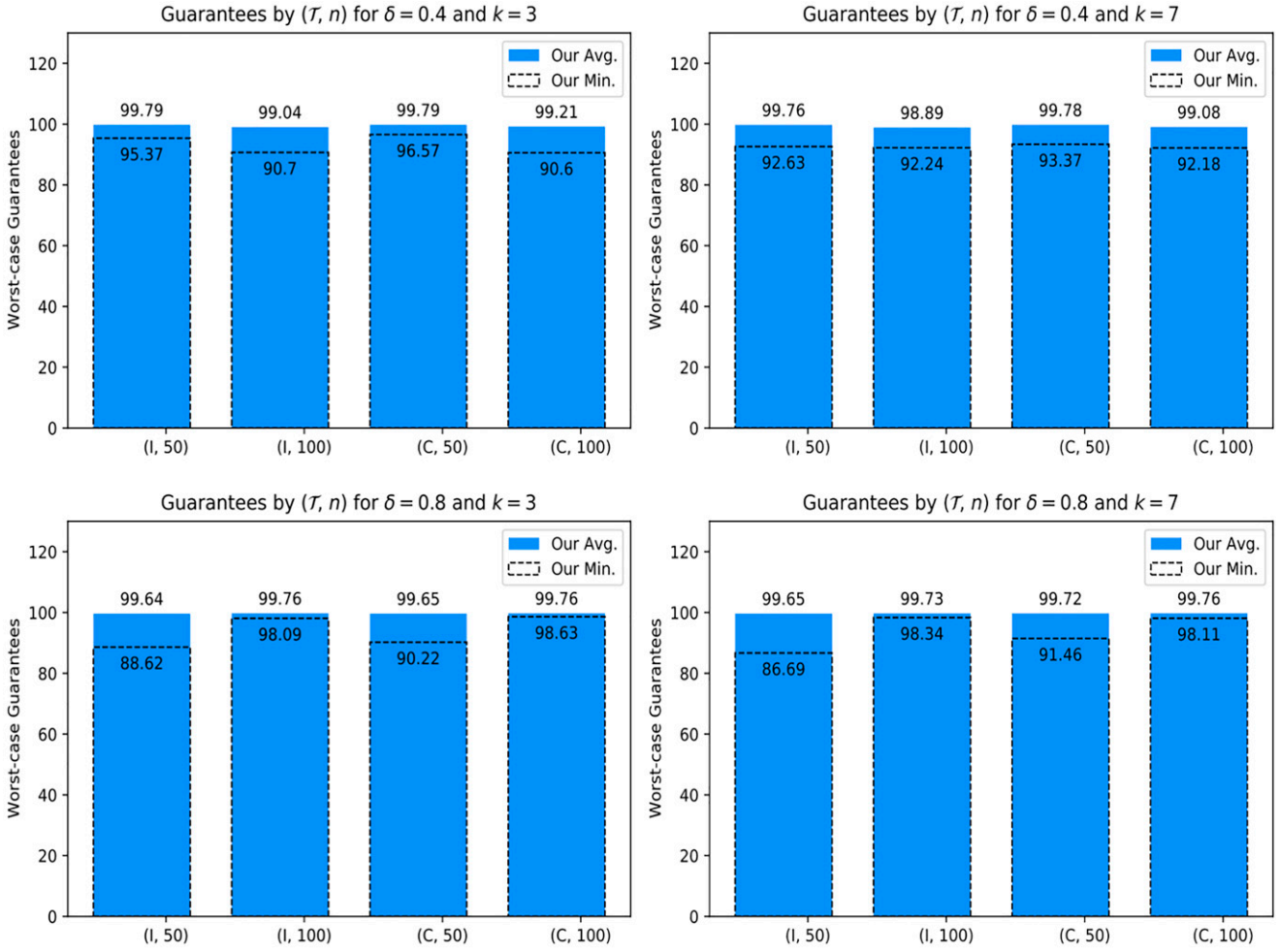
7.2. Reported Quantities

For conciseness, we group the results by $(T, n) \in \{I, C\} \times \{50, 100\}$. For the unconstrained assortment problem, we obtain four bar plots, which are all plotted on the same graph. See Figure 2. Each configuration of (T, n) involves nine configurations of $(\overline{\gamma}, P_0)$. Over these 900 instances, we record the average performance and the worst-case performance. We use three graphs in the cardinality-constrained case, wherein each graph corresponds to a value of $\delta \in \{0.2, 0.5, 0.8\}$. As in the unconstrained case, the configurations are grouped by (T, n) values. Detailed computational tables can be found in Online Appendix F.

7.3. Comparison with Zhang et al. (2020)

For the unconstrained and cardinality-constrained cases, we also compare our approach with that of Zhang et al. (2020) by running both algorithms on the same instances. We summarize the results in Figure 2. In the unconstrained case, our overall average performance is ~ 1% better than that of Zhang et al. (2020), whereas we see an improvement of about 3% in the worst-case performance. The differences are more pronounced in the cardinality-constrained instances. For $\delta = 0.2$ and 0.8, the difference in average performance is similar to the unconstrained case but the worst-

Figure 4. (Color online) Summary of Results for Partition Constrained Assortment Optimization



case guarantees show a larger gap; for example, we observe ~8% difference in the worst-case for group $(C, 100)$ when $\delta = 0.8$. On configurations with $\delta = 0.5$, our algorithm outperforms Zhang et al. (2020) by over 10% on average and by almost 15% in the worst case.

7.4. Knapsack and Partition Constraints

Test instances for knapsack constraints are labeled $(T, n, \overline{\gamma}, P_0, \eta)$. Here, the capacity is set to one and $\eta \in \{0.1, 0.25, 0.5, 1\}$, where each product's size is chosen uniformly from $[0, \eta]$. Figure 3 summarizes our results for knapsack constraints, where we divide the configurations into four sets based on η values. Test instances for partition constraints are labeled $(T, n, \overline{\gamma}, P_0, \delta, k)$, where $k \in \{3, 7\}$ is the number of parts and products are assigned randomly to the k parts. Parameter $\delta \in \{0.4, 0.8\}$ indicates the capacity of each part: if there are p products in a part, then its capacity is set to δp. Figure 4 summarizes the results for partition constraints: we divide the configurations into four sets based on the (δ, k) values.

The results for knapsack constraints vary based on the value of η. Although the average performance is greater than 92% for all groups, we observe a worst case of ~59% when $\eta = 1$. Our approach performs extremely well in the case of partition-constrained assortment optimization. The average, in all cases, is over 99%, whereas the worst-case over all tests is 86%.

7.5. Computational Results for the PTAS

We also test our randomized PTAS for unconstrained assortment optimization on moderately sized instances. We use the same instance-generation described earlier (for the unconstrained problem) but with number of products $n \in \{30, 50\}$. We set parameter $\epsilon = 0.1$ in the PTAS (Algorithm 3). For this choice of ϵ and instance size ($n \leq 50$), the sampling probability p (in Algorithm 3) turns out to be one, which would lead to complete enumeration. Instead, we reduce the sampling probability p to 0.15 (for $n = 30$) and 0.1 (for $n = 50$). Note that reducing p only worsens the algorithm's approximation guarantee. Finally, for each instance,

we repeat the randomized PTAS three times and return the best solution. By executing these runs in parallel, one could potentially reduce the runtime by a factor of ~ 3. We observe an average performance ratio of 99% and worst-case performance ratio of 84%. The worst-case runtime is less than 30 seconds for $n = 30$ and ~ 90 seconds for $n = 50$.

8. Conclusion

We demonstrated a close connection between the assortment optimization problem under the PCL model and max-dicut. Combined with good convex programming relaxations and new LP-rounding algorithms, we obtained significantly improved approximation algorithms for the unconstrained, cardinality-constrained, and knapsack-constrained assortment problems. Furthermore, we designed a new approximation framework for this class of assortment optimization problems that bypasses the need for convex relaxations. This enabled us to handle more general constraints, such as a multidimensional knapsack constraint and (nested) partition constraints. Using additional properties of the PCL model and the relation to max-dicut, we also obtained a randomized PTAS for the unconstrained assortment problem and the cardinality-constrained problem when the capacity is not too small. Obtaining similar improvements for the other constrained versions is an interesting open question.

Acknowledgments

The authors thank the three referees and associate editor for several helpful comments and suggestions that improved this paper. In particular, their suggestion to further investigate the structure of the max-dicut instances lead us to obtain Theorem 10.

References

Ageev AA, Hassin R, Sviridenko M (2001) A 0.5-approximation algorithm for MAX DICUT with given sizes of parts. *SIAM J. Discrete Math.* 14(2):246–255.

Ageev AA, Sviridenko M (1999) Approximation algorithms for maximum coverage and max cut with given sizes of parts. Cornuejols G, Burkard RE, Woeginger GJ, eds. *Integer Programming Combinatorial Optimization*, Lecture Notes in Computer Science, vol. 1610 (Springer), 17–30.

Arora S, Karger DR, Karpinski M (1999) Polynomial time approximation schemes for dense instances of np-hard problems. *J. Comput. System Sci.* 58(1):193–210.

Bekhor S, Prashker J (1998) Investigation of stochastic network loading procedures. *Transportation Res. Record* 1645(1):94–102.

Blanchet J, Gallego G, Goyal V (2016) A Markov chain approximation to choice modeling. *Oper. Res.* 64(4):886–905.

Buchbinder N, Feldman M (2019) Constrained submodular maximization via a nonsymmetric technique. *Math. Oper. Res.* 44(3): 988–1005.

Chen A, Ryu S, Xu X, Choi K (2014) Computation and application of the paired combinatorial logit stochastic user equilibrium problem? *Comput. Oper. Res.* 43(1):68–77.

Davis J, Gallego G, Topaloglu H (2013) Assortment planning under the multinomial logit model with totally unimodular constraint structures. Working paper, Cornell University, Ithaca, NY.

Davis J, Gallego G, Topaloglu H (2014) Assortment optimization under variants of the nested logit model. *Oper. Res.* 62(2): 250–273.

Feldman J (2017) Technical note: Space constrained assortment optimization under the paired combinatorial logit model. Preprint, submitted August 5, https://dx.doi.org/10.2139/ssrn.3013321.

Feldman J, Topaloglu H (2015) Capacity constraints across nests in assortment optimization under the nested logit model. *Oper. Res.* 63(4):812–822.

Gallego G, Topaloglu H (2014) Constrained assortment optimization for the nested logit model. *Management Sci.* 60(10):2583–2601.

Goemans M, Williamson D (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* 42(6):1115–1145.

Håstad J (2001) Some optimal inapproximability results. *J. ACM* 48(4):798–859.

Karoonsoontawong A, Lin DY (2015) Combined gravity model trip distribution and paired combinatorial logit stochastic user equilibrium problem. *Networks Spatial Econom.* 15(4):1011–1048.

Khot S (2002) On the power of unique 2-prover 1-round games. *Proc. 34th Annual ACM Sympos. Theory Comput.* (ACM, New York), 767–775.

Khot S, Kindler G, Mossel E, O'Donnell R (2007) Optimal inapproximability results for max-cut and other 2-variable CSPS? *SIAM J. Comput.* 37(1):319–357.

Koppleman F, Wen CH (2000) The paired combinatorial logit model: Properties, estimation and application. *Transportation Res. B: Methodological* 34(2):75–89.

Kulik A, Shachnai H, Tamir T (2013) Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Math. Oper. Res.* 38(4):729–739.

Lee J, Mirrokni VS, Nagarajan V, Sviridenko M (2010) Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM J. Discrete Math.* 23(4):2053–2078.

Lewin M, Livnat D, Zwick U (2002) Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. Cook WJ, Schulz AS, eds. *Integer Programming Combinatorial Optim.*, Lecture Notes in Computer Science, vol. 2337 (Springer, Berlin, Heidelberg), 67–82.

Li H, Webster S (2017) Optimal pricing of correlated product options under the paired combinatorial logit model. *Oper. Res.* 65: 1215–1230.

McFadden D (1974) Conditional logit analysis of qualitative choice behavior. Zarembka P, ed. *Frontiers in Economics* (Academic Press, New York), 105–142.

Rusmevichientong P, Shen ZJM, Shmoys D (2010) Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Oper. Res.* 58(6):1666–1680.

Rusmevichientong P, Shmoys D, Tong C, Topaloglu H (2014) Assortment optimization under the multinomial logit model with random choice parameters. *Production Oper. Management* 23(11): 2023–2039.

Talluri K, van Ryzin G (2004) Revenue management under a general discrete choice model of consumer behavior. *Management Sci.* 50(1):15–33.

Zhang H, Rusmevichientong P, Topaloglu H (2020) Assortment optimization under the paired combinatorial logit model. *Oper. Res.* 68(3):741–761.

Rohan Ghuge is a PhD candidate in industrial and operations engineering at the University of Michigan. Previously, he obtained his masters in computer and information science at the University of Pennsylvania. Rohan's research

interests include combinatorial and stochastic optimization, approximation algorithms, and problems in network design.

Joseph Kwon has a bachelor's degree in industrial and operations engineering from the University of Michigan (2019). His research interests are in applied optimization problems.

Viswanath Nagarajan is an associate professor in industrial and operations engineering at the University of Michigan, where he has been since 2014. Viswanath's research interests are in combinatorial optimization, approximation algorithms, and stochastic optimization.

Adetee Sharma is a graduate from the University of Michigan, where she obtained her MS in industrial and operations engineering. Currently, she is working as a data scientist at Axtria, a pharmaceutical consulting firm. Adetee's main areas of focus are specialty pharmacy analytics, product management, and patient journey optimization.