Constrained Assortment Optimization under the Paired Combinatorial Logit Model

Rohan Ghuge, Joseph Kwon, Viswanath Nagarajan, Adetee Sharma
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109

We study the assortment optimization problem when customer choices are governed by the paired combinatorial logit model. We study unconstrained, capacitated 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 (PTAS) for the unconstrained version and performance guarantees of 50% and $\approx 50\%$ for the capacitated 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 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. Our computational results are very good, demonstrating much better empirical performance than the above-mentioned worst-case bounds.

Key words: assortment optimization, submodularity, linear and semidefinite programming

History: This paper was first submitted on ...

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 she is interested in buying. If that product is offered, then she makes the purchase. If it's not offered, then she 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 these problems. In addition, we provide the first constant-factor approximation algorithms for PCL assortment optimization under other natural constraints such as multidimensional knapsack and partition constraints. These constraints are significantly harder to handle and have not been studied previously. In the multidimensional-knapsack version, each product has multiple attributes and there is a constraint on the total size of each attribute. In the partition version, products are partitioned into different categories and there is a limit on the number of selected products in each category.

As an example of multidimensional-knapsack constraints, consider a supermarket like Fred Meyer or Kroger where products have a number of physical attributes: weight, volume, width, etc. When stocking products on a shelf, there is a limit on the total value of each individual attribute that cannot be exceeded. In order to ensure that all such constraints are respected, we can introduce a multidimensional knapsack constraint, one dimension for each attribute.

Assortment optimization and pricing problems under the multinomial logit (MNL) and nested logit (NL) models have been extensively studied. In the multinomial logit model, the probability

of purchasing a product is a ratio of the preference weight of the product to the sum of preference weights of the offered products and the preference weight of a 'no-purchase' option (which captures the situation when the customer does not make a purchase). The MNL model suffers from the independence from irrelevant alternatives (IIA) property, where the relative probability of someone choosing between two options is independent of any additional alternatives in the choice set. We point the interested reader to McFadden (1976) and Green and Srinivasan (1978) for some intuitive counterexamples that do not satisfy the IIA property. One way to combat the IIA property is to use the nested logit model. The NL model allows correlation among some choices: products in the same nests are correlated while those belonging to different nests are independent. However, the types of correlations that the NL model captures is rather restrictive.

The PCL model is a different choice model that has been gaining popularity in the assortment literature, as seen in the works of Feldman (2017) and Zhang et al. (2020). This is a random utility model, similar to the widely-used MNL and NL models (McFadden (1974)). It is a member of the generalized extreme value (GEV) family of models, and is consistent with random utility maximization when the dissimilarity parameters between every pair of products (mentioned later) lie in the unit interval. In fact, the MNL model is a special case of the PCL model. Koppleman and Wen (2000) studied the correlations among utilities in the PCL model versus that in MNL and NL models. They also argued that compared to NL models, estimating the parameters for the PCL model is advantageous because we do not need to search among numerous NL nesting structures.

Another advantage over the MNL and NL models, is that the PCL model provides a general correlation structure among the utilities of different products. There are parameters for every pair $(i; j)$ of products that capture the dissimilarity between products i and j. So, the PCL model can help capture certain phenomena of customer choice that cannot be modeled by the MNL or NL models. Firstly, utilities of different products may be arbitrarily correlated. The MNL model assumes that the utilities of products are uncorrelated while the NL model assumes equal correlation among the utilities of the products in one nest. However, the PCL model can have a different correlation

coefficient between the utilities of each pair of products. This helps to model the situation when a higher utility for one product implies a high utility for a similar product. Secondly, the PCL model can capture the situation in which the correlation structure of the utilities does not satisfy the "transitivity" property. For example, it could be possible that the utility of product 1 is correlated to the utilities of products 2 and 3, but the utilities of products 2 and 3 are uncorrelated. The MNL and NL models also fail to capture this situation. We refer the interested reader to Zhang et al. (2020) for a more thorough discussion on the correlation structure provided by the PCL model. Zhang et al. (2020) also conducted a numeric study fitting the PCL and MNL models to a real dataset involving hotel room purchases, and found that the PCL model fit the data better.

1.1. Our Contributions

First, we establish a novel connection from the 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 non-negative weights).

Second, we combine the above insight with the approximation framework of Zhang et al. (2020) and new linear-program rounding algorithms to obtain improved approximation algorithms for the unconstrained, capacitated 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 capacitated version, we obtain an approximation guarantee of 0.5 using a linear program (LP) 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 a capacity constraint. While a 0.5-approximation algorithm for max-dicut with capacity constraints can be obtained by directly using the algorithm from (Ageev et al. (2001)), such an approach would involve solving up to n LPs. Instead, we obtain an algorithm that only requires one LP. For the knapsack-constrained version, we obtain an approximation guarantee of 0.25. This also uses an LP relaxation, but the rounding requires extending the pipage rounding method for max-dicut from cardinality to general knapsack constraints. We thus improve previously known approximation guarantees for all the basic versions of PCL assortment optimization: unconstrained (from 0.79 to 0.874), capacitated (from 0.25 to 0.5) and knapsack-constrained (from 0.083 to 0.25). For the capacitated 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 (Zhang et al. (2020)). Although we use the approximation framework for PCL assortment optimization from Zhang et al. (2020), our LP/SDP relaxations are different from those used before. Moreover, our rounding algorithms are completely different.

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. To the best of our knowledge, previous papers did not consider any other constraints on feasible assortments. Indeed, an important limitation of the approximation framework in Zhang et al. (2020) is that it requires good LP (or SDP) relaxations that handle the underlying constraint. This makes their framework inapplicable (or hard to apply) to more general constraints such as a multidimensional knapsack constraint and partition constraints. We show that the assortment optimization problem under any constraint $\mathcal{C}$ admits an ($_{\mathcal{C}}$)-approximation algorithm where $_{\mathcal{C}}$ is the approximation guarantee for max-dicut under the same constraint $\mathcal{C}$ and > 0 is any constant. Thus our framework gives an *approximation-preserving* reduction from the assortment problem to the constrained max-dicut problem. As applications, we obtain (i) an improved (0.5)-approximation algorithm for the knapsack-constrained assortment problem and (ii) the first constant-factor approximation

algorithms for the assortment problem under matroid and multi-dimensional knapsack constraints. The first application relies on an improved (0.5)-approximation algorithm for max-dicut under a knapsack constraint that we obtain in this paper; the running time is $n^{O(1/)}$. 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 out-going 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 non-negative 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 > 0, our algorithm provides a (1) approximate solution with high probability in $n^{O(1/ ^2)}$ 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 an FPTAS for this problem. We also obtain a randomized PTAS for the capacitated problem when the capacity is not too small, namely $c = \Omega(n)$.

Constraint type	**Max-dicut**	**PCL-AO**
Unconstrained	0.874 (Lewin et al. (2002))	**1** (§6)
Capacitated	0.5 (Ageev et al. (2001))	**0.5** (§4.6) **1** if $c = \Omega(n)$ (§6)
Knapsack	**0.5** (§5.1)	**0.5** (§5.1)
Partition / Multi-dimensional knapsack	0.385 (Buchbinder and Feldman (2019))	**0.385** (§5.2)

Table 1 Summary of results: max-dicut v/s PCL Assortment Optimization (our results in bold).

Fifth, we perform extensive computational experiments on previously-used test instances from Zhang et al. (2020), and observe that our algorithms perform very well. For all of our experiments, we compute an upper-bound for the optimal value of the given assortment problem using an LP (see §7 for more details). For both the unconstrained and capacitated versions, our empirical performance is about 99% on average. For the knapsack-constrained assortment optimization problem, our average performance is above 94%. 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).

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

There has been a fair amount of work on the PCL model in the past few years. Li and Webster (2017) studied pricing problems under the PCL model. Zhang et al. (2020) studied the assortment optimization problem under the PCL model. The authors studied two variants - the unconstrained and capacitated assortment optimization. They proved that even the unconstrained version is NP-hard and gave a framework for designing approximation algorithms for PCL assortment optimization. Using LP relaxations, they obtained a 0.6-approximation guarantee for the unconstrained version and a 0.25-approximation guarantee for the capacitated version. Using SDP techniques, they

achieved an improved 0.79-approximation guarantee for the unconstrained version. Feldman (2017) studied the knapsack-constrained assortment optimization problem under the PCL model. Using techniques developed by Feldman and Paul (2018) and Zhang et al. (2020), the author obtained a 0.083 approximation guarantee for this problem. We also use the approximation framework from Zhang et al. (2020) in designing better approximation algorithms for all three versions discussed above. Our main new ingredients are the connection to max-dicut, different SDP/LP relaxations and their rounding algorithms.

These works show that research in solving operational problems under the PCL model is gaining popularity. There already exists a considerable amount of work in the transportation literature that uses the PCL model to solve problems as in the works of Chen et al. (2014), Karoonsoontawong and Lin (2015), and Bekhor and Prashker (1998). Koppleman and Wen (2000) and Bekhor and Prashker (1998) showed that the PCL model is statistically superior to the MNL and NL models in the context of modeling route choices. There is a complex correlation structure among the utilities obtained from different routes since different routes can overlap with each other at various roads. This correlation structure is captured well by the PCL model. Chen et al. (2014) and Karoonsoontawong and Lin (2015) considered various traffic equilibrium problems and discussed the benefits of using the PCL model as a solution concept for these problems. This research shows that the PCL model outperforms the MNL and NL models in predicting user behavior, especially when there exists 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. Talluri and van Ryzin (2004) showed that a nested allocation policy, nested by revenues, is optimal for assortment optimization under the MNL model. Rusmevichientong et al. (2010a) investigated the capacitated assortment optimization problem under the MNL model. They provided an efficient (polynomial time) algorithm to solve the problem. Davis et al. (2013) showed that assortment optimization under the MNL model can be directly formulated and solved as a linear program. This allowed them to solve the assortment problem with different constraints, and also solve some pricing problems under the MNL model.

Davis et al. (2014) studied the assortment optimization problem under the nested logit model. They gave conditions required for the problem to be polynomially solvable. In the absence of the aforementioned conditions, they proved the problem to be NP-hard, and designed algorithms with worst-case guarantees. Gallego and Topaloglu (2014) and Feldman and Topaloglu (2015) studied the cardinality and knapsack constrained versions of the assortment optimization problem under the nested logit model. Gallego and Topaloglu (2014) investigated the problem when the constraints are on the set of products offered in each nest, while Feldman and Topaloglu (2015) studied the problem when the constraints are on all products (in all nests). Both gave efficient algorithms for the problem under cardinality constraints. For the knapsack constrained version, Gallego and Topaloglu (2014) gave a 0.5-approximation algorithm, while Feldman and Topaloglu (2015) gave a 0.25-approximation algorithm.

Assortment optimization has also been studied under other models such as mixtures of MNL and the Markov chain choice model. See Rusmevichientong et al. (2010b) and Blanchet et al. (2016) for details. Rusmevichientong et al. (2010b) show that the assortment optimization problem under a mixture of MNL models is NP-hard, and provide the first polynomial-time approximation scheme (PTAS) for the problem while Blanchet et al. (2016) design an efficient optimal 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. Linear program relaxations can be used to achieve 0.5-approximation algorithms for 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 0.859 by Fiege and Goemans (1995), and further to 0.874 by Lewin et al. (2002). Max-cut is NP-hard; in fact, assuming the unique-games-conjecture of Khot (2002), Khot et al. (2007) showed that one cannot obtain any approximation ratio better than 0.878 for this problem. 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 (this is discussed in more detail later), they developed a 0.5 approximation algorithm for this problem.

1.3. Organization

In §2, we set up the notation, and formulate all the versions of the assortment optimization problem. In §3, we describe our key reduction from the assortment optimization problem to the max-dicut problem. In §4, we first review the requisite results from Zhang et al. (2020); then we present our new algorithms for the unconstrained, capacitated and knapsack-constrained versions. In §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 §6, we obtain the randomized PTAS for the unconstrained assortment optimization problem. We present computational results in §6 and conclude in §7. All missing proofs appear in the appendix.

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 $\mu_{ij} \in [0, 1]$ be the dissimilarity parameter of that nest. 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 1 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 $V_{ij}(\mathbf{x})^{\mu_{ij}}$, where $V_{ij}(\mathbf{x}) = v_i^{1-\mu_{ij}} x_i + v_j^{1-\mu_{ij}} x_j$. Thus, given $\mathbf{x}$, the probability that a customer picks nest (i, j) is

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

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

$$\frac{v_i^{1=\ ij}x_i}{v_i^{1=\ ij}x_i + v_j^{1=\ ij}x_j};$$

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 $(i;j)$, and by the law of total expectation, we have

$$R_{ij}(\mathbf{x}) = \frac{r_i v_i^{1=\ ij}x_i + r_j v_j^{1=\ ij}x_j}{v_i^{1=\ ij}x_i + v_j^{1=\ ij}x_j} = \frac{r_i v_i^{1=\ ij}x_i + r_j v_j^{1=\ ij}x_j}{V_{ij}(\mathbf{x})}:$$

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

$$(\mathbf{x}) = \sum_{(i;j)2M} \frac{V_{ij}(\mathbf{x})^{\ ij}}{v_0 + \sum_{(k;l)2M} V_{kl}(\mathbf{x})^{\ kl}} R_{ij}(\mathbf{x}) = \frac{\sum_{(i;j)2M} V_{ij}(\mathbf{x})^{\ ij} R_{ij}(\mathbf{x})}{v_0 + \sum_{(k;l)2M} V_{kl}(\mathbf{x})^{\ kl}}: \tag{1}$$

We note that the PCL model is usually defined using *unordered* pairs $M^0 = f(i;j)j1 \quad i < j \quad ng$ as the nests. In this paper, we use a definition with ordered pairs because it simplifies notation and it still captures the usual model. Indeed, given an instance of the usual PCL model (where $_{ij}$ is defined for unordered pairs), we set $_{ij} = \ _{ji}$ for all $i;j \ 2 \ [n]$ and scale the preference weight of the no-purchase option by two (i.e., set $v_0^0 = 2v_0$). 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:

$$z \ = \ (\mathbf{x}\ ) = \max_{\mathbf{x}2F} \ (\mathbf{x}); \tag{2}$$

where $F \quad f0;1g^n$ is the set of all feasible subsets of products. Throughout this paper, we assume that F is *downward closed*, that is, for any $A \ 2 \ F$ and $B \quad A$ we also have $B \ 2 \ F$. We consider the following specific constraints:

Unconstrained: $F = f0;1g^n$.

Capacitated: $F = f\mathbf{x} \ 2 \ f0;1g^n \ j \ \sum_i x_i \quad cg$ where c is the limit on the number of products.

Knapsack constrained: $F = \{\mathbf{x} \in \{0,1\}^n : \sum_i s_i x_i \le c\}$ where each s_i represents the size of product i, and c is the limit on total size.

Multidimensional knapsack constraint: $F = \{\mathbf{x} \in \{0,1\}^n : \sum_{i=1}^{n} s_{ki} x_i \le c_k \ \ k = 1,2,\ldots,q\}$ which involves q different knapsack constraints. Here we assume that q is a constant.

Partition: $F = \{\mathbf{x} \in \{0,1\}^n : \sum_{i \in I_k} x_i \le c_k, \ k = 1,2,\ldots,q\}$ 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

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

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

Definition 1 (Maximum Directed Cut (max-dicut)). 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 out-going edges, i.e.,

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

It is usually assumed that all edge weights are non-negative.

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 $[n] := \{1, 2, \ldots, n\}$ represent the products and vertex $n+1$ is a dummy vertex. We associate a binary decision variable x_i with each of the vertices $i \in \{1, 2, \ldots, n+1\}$ which represents whether/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 $\bar{r}_i = r_i - z$ 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

$$\alpha^+_{ij} = \frac{\bar{r}_i v_i^{1-\mu_{ij}}}{(v_i^{1-\mu_{ij}} + v_j^{1-\mu_{ij}})^{1-\mu_{ij}}} \quad \text{and} \quad \alpha_{ji} = \frac{\bar{r}_j v_j^{1-\mu_{ij}}}{(v_i^{1-\mu_{ij}} + v_j^{1-\mu_{ij}})^{1-\mu_{ij}}}. \tag{4}$$

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

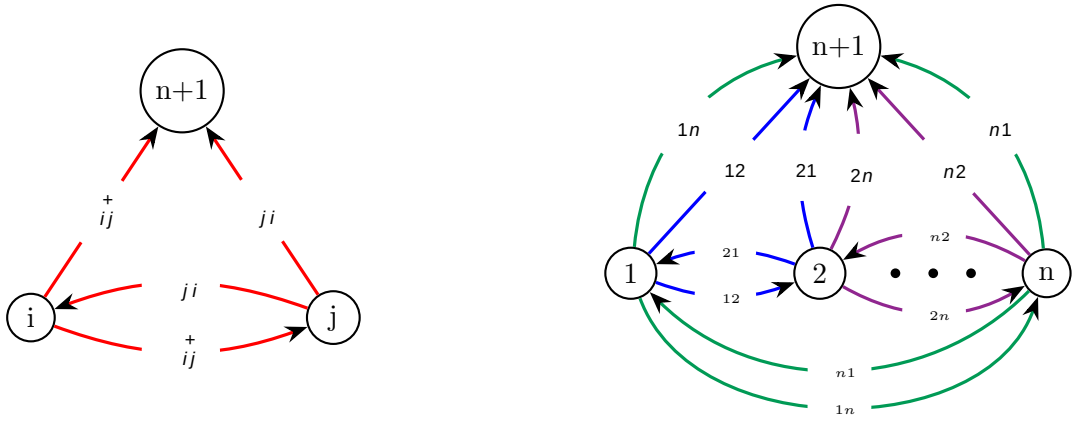
$$\beta^+_{ij} = v_i - \bar{r}_i \quad \beta^+_{ij} = v_i - \bar{r}_i - (1-\mu_{ij}) \frac{v_i^{1-\mu_{ij}}}{(v_i^{1-\mu_{ij}} + v_j^{1-\mu_{ij}})^{1-\mu_{ij}}} \quad \text{and} \quad \beta_{ji} = v_j - \bar{r}_j \quad \beta_{ji} = v_j - \bar{r}_j - (1-\mu_{ij}) \frac{v_j^{1-\mu_{ij}}}{(v_i^{1-\mu_{ij}} + v_j^{1-\mu_{ij}})^{1-\mu_{ij}}}. \tag{5}$$

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

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

where $\alpha_{ij} = \alpha^+_{ij} + \alpha_{ij}$, and $\beta_{ij} = \beta^+_{ij} + \beta_{ij}$. 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 $[n+1] := \{1, 2, \ldots, n, n+1\}$ (also denoted by V) and the edge-set is denoted E.



(a) Edges added for pair $(i;j)$. (b) Overview of all edges.

Figure 1 Structure of the graph G_z.

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 $r_i < 0$ (resp. $r_i \geq 0$) for a vertex i, then every edge leaving vertex i in G_z has a negative (resp. non-negative) weight.*

For ease of notation, we define the following two sets: $N_z := \{i \in [n] \mid r_i < 0\}$ and $P_z := V \setminus N_z$. Note that N_z consists of those vertices that have negative weight out-going edges. The dummy vertex has no edges leaving it, and thus is in the set P_z. Let subgraph $\hat{G}_z$ consist of all non-negative weight edges in G_z. Note that $\hat{G}_z$ 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:

$$\text{maximize:} \sum_{(i;j)\in M} \frac{r_i v_i^{1=\,ij} x_i + r_j v_j^{1=\,ij} x_j}{V_{ij}(\mathbf{x})^{1\quad ij}} \qquad \mathbf{x} \in F$$

$$\text{subject to: } \mathbf{x} \in \{0;1\}^n \qquad (\mathrm{IP}_1(z)) \qquad \text{maximize:} \sum_{(i;j)\in E} w_{ij}(z)\ y_{ij}$$

$$\begin{aligned}
&\text{subject to: } y_{ij} \quad x_i; && 8(i;j) \ 2 \ E && x_i \ 2 \ f0;1g; && 8i \ 2 \ V \\
&\qquad y_{ij} \quad 1 \quad x_j; && 8(i;j) \ 2 \ E && x_{n+1} = 0 \\
&\qquad y_{ij} \ 2 \ f0;1g; && 8(i;j) \ 2 \ E \quad (\mathrm{IP}_2(z)) && \mathbf{x} \ 2 \ F
\end{aligned}$$

where F represents the set of feasible solutions to the assortment problem. Note that a feasible solution for $\mathrm{IP}_2(z)$ is $(\mathbf{x}; x_{n+1}) \ 2 \ F \quad f0g$. We now argue that there are optimal solutions for $\mathrm{IP}_1(z)$ and $\mathrm{IP}_2(z)$ such that no $i \ 2 \ N_z$ is included in the optimal assortment or in the optimal cut.

Lemma 3. *Given an optimal solution* $\mathbf{x}$ *to* $IP_1(z)$*, we can construct another optimal solution* $\overline{\mathbf{x}}$ *such that for all* $i \ 2 \ N_z$*,* $\overline{x}_i = 0$*, i.e. no product with* $\overline{r}_i < 0$ *contributes to the solution, and for all* $i \ 2 \ P_z$ *(excluding* $n+1$*, since there are only* n *items),* $\overline{x}_i = x_i$*.*

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* $i \ 2 \ N_z$*,* $\overline{x}_i = 0$*, i.e. no vertex that has outgoing edges of negative weight contributes to the solution, and for all* $i \ 2 \ P_z$*,* $\overline{x}_i = x_i$*.*

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* F*.*

Note also that both $\mathrm{IP}_1(z)$ and $\mathrm{IP}_2(z)$ are functions of z.

Lemma 6. *The formulations* $IP_1(z)$ *and* $IP_2(z)$ *are equivalent, that is, for all* $z \ 2 \ \mathrm{R}$*, the optimal value of* $IP_1(z)$ *is the same as the optimal value of* $IP_2(z)$*. Moreover, for each* $z \ 2 \ \mathrm{R}$*, the optimal value of* $IP_2(z)$ *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* F*.*

Complexity of the reduction: We note that even when the input to the assortment problem (values r_i, v_i and $_{ij}$) is rational, the reduction above may not be polynomial sized in the input, since the weights $w_{ij}(z)$ may be irrational. As we saw in the proof of Lemma 2, we can write $w_{ij}(z) = \overline{r}_i \quad _{ij}$ where $_{ij} \quad 0$. In Appendix A.1, we give an approach to deal with irrational $_{ij}$

values using Dirichlet's approximation theorem. The idea is to compute rational values $\overline{\ }_{ij}$ with polynomial bit complexity such that $j \ _{ij} \quad \overline{\ }_{ij} j \quad$ for $= \frac{1}{r_{\max} n^3}$. Then, we use $\overline{w_{ij}}(z) = \overline{\ }_{ij}(r_i \quad z)$ 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.

4. Unconstrained, Capacitated 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 $g(z) \quad f(z)$ for all $z \ 2 \ \mathsf{R}$.
2. Find the fixed point $\hat{z}$ of $g(\ ) = v_0$; i.e., find a $\hat{z} \quad 0$ such that $g(\hat{z}) = v_0 \hat{z}$.
3. Find an assortment $\hat{\mathbf{x}} \ 2 \ F$ such that $\sum_{(i;j) \ 2 \ M} V_{ij}(\hat{\mathbf{x}}) \ _{ij}(R_{ij}(\hat{\mathbf{x}}) \quad \hat{z}) \quad g(\hat{z})$ for some $\quad 2 \ [0; 1]$.

Theorem 1 (**Theorem 3.1 from Zhang et al. (2020)**). *Assume that we have an algorithm that satisfies steps 1-3 mentioned above. Then, we have* $(\hat{\mathbf{x}}) \quad \hat{z} \quad z$ *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 approximate solution to the assortment optimization problem.*

To use Theorem 1, 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 which also determines the approximation ratio . Our main contribution is in this step.

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

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

4.1. Relating $f(z)$ to max-dicut

We now relax $\mathrm{IP}_2(z)$ into a linear program by letting variables take values in $[0,1]$, and by explicitly stating the constraint for $\mathbf{x} \in F$.

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

Clearly, $g(z)$ is an upper bound to $f(z)$ for all $z \in \mathbb{R}$, as required in Step 1 of Theorem 1. 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 $i \in N_z$, $\overline{x}_i = 0$, i.e. 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 $\mathcal{G}_z$. Note that $\mathcal{G}_z$ has non-negative edge weights, and hence, will be a valid input for the approximation algorithms used later. Moreover, due to Lemma 6, a maximum directed cut in $\mathcal{G}_z$ 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. $g(z)$ is non-increasing 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 computation of the fixed point in §4.2 and the overall algorithm in §4.3. The unconstrained problem is discussed in §4.4. We then provide some preliminary results on the LP-rounding approach that we use in §4.5. Finally, §4.6 addresses the capacitated and knapsack-constrained problems.

4.2. Fixed point for g(z)

Continuing with the approach from Theorem 1, 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 for the constraints in LP(z).

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

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:

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

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

4.3. The Overall Approach

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 $\mathcal{G}_z$ with *non-negative* edge-weights.

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

4.4. The Unconstrained Problem

For the unconstrained version, we use a semi-definite program (SDP) relaxation as the upper-bound $g(z)$. The overall algorithm remains the same as in §4.3, except for the use of SDPs in steps 1-2 and an SDP-based lemma (similar to Lemma 7) that ensures that we can focus on non-negative 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 Appendix B).

Theorem 2. *There is a randomized* 0.874*-approximation algorithm for the unconstrained PCL assortment optimization problem.*

4.5. 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 capacitated 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) below.

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

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

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, i.e., $\sum_i s_i x_i = c$.

We define two properties:

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

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

We now describe the pipage rounding procedure. Let $\bar{\mathbf{x}}$ be any solution to LP (8) that satisfies the condition in Definition 2. Starting with $\mathbf{x} = \bar{\mathbf{x}}$, 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, i.e. $0 < x_i, x_j < 1$.
2. By the convexity property, we have $(\mathbf{x}, i, j, \ ) \ge F(\mathbf{x})$ for either $\ = -\min\{x_i, \frac{s_j}{s_i}(1 - x_j)\}$ or $\ = \min\{(1 - x_i), \frac{s_j}{s_i} x_j\}$. Let $\ $ denote the value so that $(\mathbf{x}, i, j, \ ) \ge F(\mathbf{x})$.
3. Obtain a new solution, $\hat{\mathbf{x}} = (x_1, \ldots, x_i + \ , \ldots, x_j - \frac{s_i}{s_j} \ , \ldots, x_n)$. Note that $F(\hat{\mathbf{x}}) \ge F(\mathbf{x})$ and either $\hat{x}_i \in \{0,1\}$ or $\hat{x}_j \in \{0,1\}$.
4. Update $\mathbf{x} \leftarrow \hat{\mathbf{x}}$.

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, i.e. $\sum_i s_i x_i = \sum_i s_i \bar{x}_i$. Let $\mathbf{x}'$ denote the final solution; note that $F(\mathbf{x}') \ge F(\bar{\mathbf{x}})$. 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 $\bar{\mathbf{x}}$ (i.e. $F(\bar{\mathbf{x}}) \ge L(\bar{\mathbf{x}})$) and $F(\mathbf{x}') \ge F(\bar{\mathbf{x}})$, we have $F(\mathbf{x}') \ge L(\bar{\mathbf{x}})$. To summarize:

Theorem 3. *Consider an instance of a problem described by* (7) *with a \nice" relaxation* (8) *and any solution* $\mathbf{x}$ *to* (8)*. Suppose that* F *satis es the convexity property and the F/L lower bound property for* $\mathbf{x}$ *and some* < 1*. Then we can nd a solution* $\mathbf{x}$ *with at most one fractional variable such that* $F(\mathbf{x}) \quad F(\mathbf{x}) \quad L(\mathbf{x})$*. 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.*

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:

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

Note that this LP is equivalent to (LP(z)), when we set $w_{ij} = w_{ij}(z)$. Moreover, as mentioned in §4.3, it suffices to consider instances with non-negative 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 4. *Let* (x, y) *be any basic feasible solution to the LP relaxation* (9)*. Then there is some value* $0 < \quad < \frac{1}{2}$ *such that* $x_i \in \{0, \ , \frac{1}{2}, 1 \quad , 1\}$ *for all* $i \in V$*. Moreover, if* $\sum_i s_i x_i < c$ *then* $x_i \in \{0, 1=2, 1\}$ *for all* $i \in V$*.*

4.6. The Capacitated 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 $F(\mathbf{x}) = \sum_{(i,j)\in E} w_{ij} x_i (1 \quad x_j)$ and $L(\mathbf{x}) = \sum_{(i,j)\in E} w_{ij} \min\{x_i, (1 \quad x_j)\}$ in (8) which is equivalent to LP (9). Note that the

convexity property (Definition 3) holds for F since the coefficient on the quadratic term in is positive. Our algorithm is now given as Algorithm 1.

Lemma 10. *The solution* $\mathbf{x}^0$ *in Step 2 satis es* $\sum_i s_i x_i^0 = \sum_i s_i x_i = c$ *and is feasible to LP* (9).

Proof For any $i \in V_1$ the increase $x_i^0 \quad x_i = \min\{1 \quad ; (1 \quad)S(V_2)/S(V_1)\} =: \Delta_1$. Similarly, for any $i \in V_2$, we have $x_i^0 \quad x_i = \quad \min\{1 \quad ; (1 \quad)S(V_1)/S(V_2)\} =: \Delta_2$. So

$$\Delta_1 \ \ S(V_1) = \min\{(1 \quad)S(V_1); (1 \quad)S(V_2)\} = \quad \Delta_2 \ \ S(V_2);$$

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

It now follows that the solution $\mathbf{x}$ obtained in Step 4 by pipage rouding (on either $\mathbf{x}$ or $\mathbf{x}^0$) also satisfies the knapsack constraint. The next two lemmas bound the objective value depending on whether/not the knapsack constraint is active. Lemma 11 generalizes a result of Ageev et al. (2001) which focused on the cardinality case (where all $s_i = 1$).

Lemma 11. *If the constraint* $\sum_i s_i x_i \quad c$ *is active then* $\max\{F(\mathbf{x}); F(\mathbf{x}^0)\} \quad 0.5 \ \ L(\mathbf{x})$.

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

We obtain the following guarantees for capacity and knapsack constraints.

Theorem 5. *Algorithm 1 gives a* 0.5*-approximation algorithm for max-dicut with a capacity constraint relative to the LP* (9)*. Hence, there is a* 0.5*-approximation algorithm for the capacitated PCL assortment optimization problem.*

Theorem 6. *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:

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 $\mathbf{x}^0$. Let $V_1 = \{i : x_i = \quad g$, $V_2 = \{i : x_i = 1 \quad g$, $V_3 = \{i : x_i = \frac{1}{2}\}$ and $V_4 = \{x_i : x_i = 0 \text{ or } 1\}$. Let $S(V_1) = \sum_{i \in V_1} s_i$, and $S(V_2) = \sum_{i \in V_2} s_i$.

$$x_i^0 = \begin{cases} \min\{1; \quad +(1 \quad)S(V_2)/S(V_1)\}; & \text{if } i \in V_1 \\ \max\{0; (1 \quad) \quad (1 \quad)S(V_1)/S(V_2)\}; & \text{if } i \in V_2 \\ x_i; & \text{if } i \in V_3 \text{ or } V_4 \end{cases} \tag{10}$$

We set $\hat{\mathbf{x}} \quad \text{argmax}\{F(\mathbf{x}); F(\mathbf{x}^0)\}$.

3: If $\sum_i s_i x_i < c$, then we set $\hat{\mathbf{x}} \quad \mathbf{x}$.

4: Apply pipage rounding (Theorem 3) to $\hat{\mathbf{x}}$ and obtain solution $\mathbf{x}$ with at most one fractional variable (say x_i). Let $I_1 = \{j : x_j = 1\}$ and $I_2 = \{i\}$.

5: **(Capacitated)** If all $s_i = 1$ then output the better of I_1 and $I_1 \cup I_2$.

6: **(Knapsack)** Otherwise, output the better of I_1 and I_2.

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

$$(i) = \sum_{(i;j) \in E} w_{ij}:$$

Corollary 1. *If $\mathbf{x}$ is an optimal solution to LP (9) and $\theta = \{j : 0 < x_j < 1\}$, Algorithm 1 obtains a solution of weight at least 0.5 LP $\max_{i \in \theta}$ (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 θ. From the proof of Theorem 6, we know that the cut value for I_1 is

$$W_{10} + W_1 \quad W_{10} + f \quad W_0 + (1 \quad f)W_1 \quad W_0 = F(\mathbf{x} \quad) \quad W_0:$$

Note that $W_0 \quad \max_{i \in \theta}$ (i). The corollary now follows as $F(\mathbf{x} \quad) \quad 0.5 \quad L(\mathbf{x}) = 0.5 \quad LP$.

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

5. Assortment Optimization under General Constraints

In this section, we provide a binary-search based approximation framework that can be used to solve PCL assortment optimization under a much larger class of constraints. It is even useful in obtaining improved approximation ratios for some simpler constraints, such as unconstrained or knapsack. Unlike the framework from Zhang et al. (2020) (Theorem 1), our approach here does not require approximation algorithms relative to LP/SDP relaxations.

Based on the "reduction" in $x3$, 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 $\mathcal{G}_z$ subject to the constraints that the dummy vertex is not selected and that the set of chosen vertices lies in F. Recall that $\mathcal{G}_z$ is the subgraph of G_z consisting of non-negative edges. So the (constrained) max-dicut instances that we need to solve will only contain non-negative edge weights. Let C be the constraint on feasible assortments (and on feasible cuts in the max-dicut instance). Let $\ _C$ denote the approximation guarantee for max-dicut under constraint C: so we can obtain an $\ _C$-approximation to $f(z)$ for any z. We use a binary search technique to obtain a value $\tilde{z}$ that is within an $\ _C$ 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 $f(\ )=v_0$. Let $\tilde{f}(z)$ be the objective value of the $\ _C$-approximation algorithm on max-dicut instance G_z with constraint C. Roughly speaking, the binary search identifies the fixed point of the approximating function $\tilde{f}(\ )=v_0$. Finally, we show how the additive error can be absorbed into the multiplicative approximation guarantee by a simple scaling idea.

Theorem 7. *Let $\ _C$ be the approximation guarantee known for the max-dicut problem under some constraint C. Then, Algorithm 2 gives an $(\ _C\ \ \ )$ approximation guarantee (for $\ >0$) for the assortment optimization problem under constraints C in time $O(\log(\ _r\ _v=\ )\ p(n))$ where $p(n)$ is the runtime of the constrained max-dicut algorithm, $\ _r=\frac{\max_i(r_i)}{\min_i(r_i)}$ and $\ _v=\frac{\max_i(v_i)}{\min_i(v_i)}$.*

5.1. Improved Algorithm for Knapsack Constraints

Here, we revisit the knapsack-constrained assortment problem where the feasible subsets are $F=\{x\in\{0,1\}^n : \sum_i s_i x_i \ \ B\}$. We obtained a 0.25-approximation algorithm for this problem in $x4.6$. We

Algorithm 2 Assortment Optimization under General Constraints

let $L \quad R_{min} := \frac{\min_i(r_i)\ \min_i(v_i)}{2\ \max_i(v_i)}$, $R \quad R_{max} := \max_i(r_i)$ and R_{min}.

while $R \quad L$ **do**

  $z \quad \frac{L+R}{2}$

  let $\hat{\mathbf{x}}$ be an $\ _C$-approximate solution to max-dicut under constraint C on graph $\mathcal{G}_z$.

  $ALG_z \quad \sum_{(i;j)2M} V_{ij}(\hat{\mathbf{x}})\ ^{ij}[R_{ij}(\hat{\mathbf{x}}) \quad z]$

  if $ALG_z \quad v_0 z$ **then** set $L \quad z$

  else Set $R \quad z$

return $\ _C$ approximation for $f(L)$ using the max-dicut reduction.

now obtain a better (0:5)-approximation algorithm for max-dicut under knapsack constraints (for any constant > 0), which combined with Theorem 7 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 §4.6. For any instance of knapsack-constrained max-dicut (with non-negative weights w_{ij}), recall from Definition 4 that (i) denotes the total weight of out-going edges from vertex $i \ 2 \ V$. Moreover, by the pipage rounding algorithm (see Corollary 1) we can obtain a solution of weight at least 0:5 $OPT \quad \max_{i2V} \ (i)$. So, if we could bound all the (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 $OPT \quad U \quad 2 \ OPT$ (we will see later how to remove this assumption). Given any > 0, set $^0 = \ =2$. Vertex $i \ 2 \ V$ is said to be *heavy* if $(i) \quad ^0U$. Let $H \quad V$ denote the set of heavy vertices. We enumerate all subsets $S \quad H$ of heavy vertices with $jSj \quad d\frac{4}{0}e$ that are feasible, i.e. $\sum_{i2S} s_i \quad c$. Note that there are at most $n^{1+4=\ ^0}$ possible subsets S: so this enumeration takes polynomial time for any constant .

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

$$x_i = \begin{cases} 1 & \text{for all } i \ 2 \ S \\ 0 & \text{for all } i \ 2 \ H \ n \ S \end{cases}$$

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 $\hat{V} = \{j \in V : 0 < x_j < 1\} \subseteq V \setminus H$ which implies $\max_{i \in \hat{V}} w(i) < \tau U$. So Corollary 1 implies that we obtain a cut of weight at least $0.5 \cdot LP(S) - \tau U$ 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 $\tau > 0$, if T denotes the set of heavy vertices with respect to τ in an optimal solution, then $|T| \leq \lceil 4/\tau \rceil$.*

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

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

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

Moreover, $\mathbb{E}[\mathrm{cut}(S)] \leq OPT$ as S is always feasible. Combining the above two inequalities, we have $OPT \geq \frac{1}{4}|T| \cdot OPT$, and rearranging we get $|T| \leq \lceil 4/\tau \rceil$.

Lastly, we need to ensure that we can find some value U with $OPT \geq U \geq \frac{1}{2} OPT$. This is done by a simple enumeration. Note that $w_{max} = \max_{(i,j) \in E} w_{ij} \leq OPT \leq \sum_{(i,j) \in E} w_{ij} \leq n^2 \cdot w_{max}$. We enumerate all values $U \in \{2^k \cdot w_{max} : k = 0, 1, \ldots, \lceil \log_2(n^2) \rceil\}$, 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 $\tau > 0$, the above algorithm has an approximation guarantee of $(0.5 - \tau)$ for max-dicut with a knapsack constraint. Hence, there is a $(0.5 - \tau)$-approximation algorithm for knapsack-constrained assortment optimization for all constant $\tau > 0$.*

We note that the LP-based approximation framework (Theorem 1) is not applicable here because our improved approximation algorithm is not relative to any LP.

5.2. Multi-Dimensional Knapsack and Matroid Constraints

It is well-known that the directed cut function on any graph with non-negative edge weights is non-negative 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 7) 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 groundset 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 non-negative submodular function under a matroid constraint, with an approximation guarantee of 0.25. The approximation ratio was improved to $\frac{1}{e}$ 0.367 by Feldman et al. (2011) and recently to 0.385 by Buchbinder and Feldman (2019); these algorithms are 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 7 we obtain:

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. To the best of our knowledge, such constraints have not been studied previously in assortment optimization.

Multidimensional knapsack constraints: Let us recall the example from the introduction. We consider the following scenario faced by a supermarket. Every product has a number of physical attributes: weight, volume, width, etc. When stocking products on a shelf, there is a limit on the total value of each individual attribute that cannot be exceeded. This situation can be modeled

by introducing a multidimensional knapsack constraint, with one dimension for each attribute. Suppose there are q attributes. Let s_{ki} denote amount of attribute k possessed by product i, and let c_k be the available limit on attribute k. Thus, the set of feasible subsets of products is $F = \{\mathbf{x} \in \{0,1\}^n : \sum_{i=1}^{n} s_{ki} x_i \leq c_k \ \ k = 1,2,\ldots,q\}$. As long as the number q of attributes is constant, we obtain a 0.385-approximation algorithm from Theorem 9.

Partition constraints: We continue with our supermarket application. A supermarket must provide a wide selection of products to its customers. There may be many available brands manufacturing a particular product-type, but obviously the supermarket cannot display all of them. For a toy example, let us consider a supermarket selling various types of bread - white, whole wheat, sour dough, etc. There are many manufacturers of these product-types but the supermarket may only have a dedicated section for each type. Thus, the supermarket must decide which brands of each product-type to put on display so as to maximize revenue. We can model this situation by introducing partition constraints. Suppose we have q different product-types, and that I_k is the set of products of type $k \in [q]$; we assume that $I_1, \ldots, I_q$ is a partition of all products. Let c_k be the limit on the number of displayed products of each type $k \in [q]$. We can then represent the set of feasible subsets as $F = \{\mathbf{x} \in \{0,1\}^n : \sum_{i \in I_k} x_i \leq c_k, \ k = 1,2,\ldots,q\}$, which is a matroid constraint. So we obtain a 0.385-approximation algorithm from Theorem 9.

Hierarchical partition constraints: A further generalization of the above application allows product-types to be *nested* and not just disjoint. Formally, if $A \subseteq [n]$ and $B \subseteq [n]$ denote the products of any two types, then we must have $A \setminus B = \emptyset$ or $A \subseteq B$ or $B \subseteq A$. The first case ($A \setminus B = \emptyset$) corresponds to disjoint types (as for partition constraints) whereas the other two cases correspond to nested types. For example, product-type "milk" may be sub-divided into the types "whole milk", "2% milk" and "1% milk". Given such a nested (or hierarchical) partition into types and a limit on the maximum number of products of each type, the goal is to select a subset of products respecting all these limits. The resulting collection F of feasible subsets is again a matroid, and 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 (PTAS) for the unconstrained assortment optimization problem under the PCL model.*

We will use the binary-search framework (Theorem 7) combined with additional properties of the max-dicut instances that arise from the assortment problem. In particular, we will show that for any $z \quad 0$, there is a randomized PTAS for the function value $f(z)$ in (3). Recall from $x3$ and Lemmas 5 and 6, that the function value $f(z)$ equals the optimal max-dicut value on $\mathcal{G}_z$ subject only to the constraint that the dummy vertex is not selected. Our focus here is the unconstrained case: so $F = f0; 1g^n$. As before, $\mathcal{G}_z$ is the subgraph of G_z consisting of non-negative edges.

In our reduction described in $x3$, recall that for every pair $(i; j) \; 2 \; [n] \quad [n]$, we add four edges: $(i; n+1)$, $(j; n+1)$, $(i; j)$ and $(j; i)$ with weights $^{+}_{ij}$; $_{ji}$; $^{+}_{ij}$, and $_{ji}$ respectively. From the definitions (4) and (5), we have $^{+}_{ij} + \; ^{+}_{ij} = v_i r_i$ and $_{ji} + \; _{ji} = v_j r_j$. Also recall the edge weights $w_{ij}(z)$ as defined in (6). In particular, for $i; j \; 2 \; [n]$ we have $w_{ij}(z) = \; ^{+}_{ij} + \; _{ij}$. Let $Q_z = fi \; 2 \; [n] : r_i \quad 0g$; recall that $P_z = Q_z \; [\; fn + 1g$ and $N_z = [n + 1] \, n \, P_z$. Let $N = jQ_z j$. For any $i \; 2 \; Q_z$ and $j \; 2 \; [n]$, we have $w_{ij}(z) \quad 2v_i r_i$ based on the calculations in Lemma 2. For easier notation, for all $i; j \; 2 \; Q_z$ we define $_{ij} = \frac{w_{ij}(z)}{v_i r_i} \; 2 \; [0; 2]$. Also, for any $i \; 2 \; Q_z$, define $C_i(z) = \; \sum_{j \, 2 \, Q_z} w_{ij}(z)$. Finally, note that the optimal max-dicut on $\mathcal{G}_z$ only contains vertices from Q_z. So $f(z)$ equals:

$$\max_{A \quad Q_z} \left(\sum_{i 2 A} v_i r_i \sum_{j 2 Q_z n A} {}_{ij} + \sum_{i 2 A} C_i(z) \right) : \tag{11}$$

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 $\overline{OPT} = Q_z \, n \, OPT$, i.e. the set of vertices that are not in the optimal max-dicut solution. For every vertex $i \; 2 \; Q_z$, define $O_i = \; \sum_{j 2 \overline{OPT}} \; _{ij}$; note that $O_i \quad 2N$. Consequently, the optimal value can be written as

$$OPT = \sum_{i 2 OPT} (v_i r_i \quad O_i + C_i(z)) :$$

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

Algorithm 3 Randomized PTAS for max-dicut on $\mathcal{G}_z$

1: Sample S: for every $i \in Q_z$, add i to S independently with probability $p = \frac{96 \log N}{\varepsilon^2 N}$.

2: If $|S| > 2Np$ then set $\hat{\mathbf{x}} = \mathbf{x}$ $= \mathbf{0}$ and return solution $\mathbf{x}$.

3: **for** all splits $(U, \overline{U})$: $U \subseteq S$, $\overline{U} = S \cap U$ **do**

4: Let $\gamma_i := \sum_{j \in \overline{U}} w_{ij}$ for all $i \in Q_z$

5: Solve the following LP and let $\hat{\mathbf{x}}$ be its optimal solution.

$$\begin{aligned} \text{maximize: } & \frac{1}{p} \sum_{i \in Q_z} v_i \gamma_i \, x_i + \sum_{i \in Q_z} C_i(z) \, x_i \\ \text{subject to: } & \sum_{j \in Q_z} w_{ij} \, (1 - x_j) \leq \frac{\gamma_i}{p} \leq N \;; \quad \forall i \in Q_z \\ & x_i \in [0, 1]; \qquad \forall i \in Q_z \end{aligned} \tag{LP( )}$$

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

We first discuss the running time of Algorithm 3. If the algorithm continues beyond step 2, the sample size $|S| \leq 2Np \leq \frac{192 \log N}{\varepsilon^2}$. 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 $N \geq \frac{3}{\epsilon}$: otherwise (11) can be easily solved by enumeration. The analysis involves showing that OPT is a feasible solution to LP( ) with high probability, showing a lower bound for OPT with respect to the fractional solution to LP( ), and finally showing that the expected objective of our rounded solution is within a factor of from OPT. We start with a key property that uses the structure of the max-dicut instance from $x3$.

Lemma 14. $OPT \geq \frac{N-1}{2} \sum_{i \in Q_z} v_i \bar{r}_i \geq \frac{N}{3} \sum_{i \in Q_z} v_i \bar{r}_i$.

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:

$$obj(R) = \sum_{i \in R} \Big(\sum_{j \in Q_z \setminus R} w_{ij}(z) + \sum_{j \in \bar{Q}_z} w_{ij}(z) \Big).$$

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,

$$\begin{aligned}
\mathsf{E}[obj(R)] &= \frac{1}{4} \sum_{i,j \in Q_z} w_{ij}(z) + \frac{1}{2} \sum_{i \in Q_z, k \notin Q_z} w_{ik}(z) \quad \frac{1}{4} \sum_{i \in Q_z} \Big(\sum_{j \in Q_z} w_{ij}(z) + w_{i,n+1}(z) \Big) \\
&= \frac{1}{4} \sum_{i \in Q_z} \Big(\sum_{j \in Q_z} w_{ij}(z) + \sum_{k \in [n] \setminus \{i\}} (\alpha^+_{ik} + \alpha_{ik}) \Big) \quad \frac{1}{4} \sum_{i,j \in Q_z} (\alpha^+_{ij} + \alpha_{ij}) + (\alpha^+_{ij} + \alpha_{ij}) \qquad (12) \\
&= \frac{N-1}{4} \sum_{i \in Q_z} 2 v_i \bar{r}_i.
\end{aligned}$$

Above, the equality in (12) uses the definition of $w_{i,n+1}(z)$ and the inequality in (12) uses the definition of $w_{i,j}(z)$ and that all the terms are non-negative. The final equality uses the observation that $\alpha^+_{ij} - \alpha^+_{ij} = v_i \bar{r}_i = \alpha_{ij} + \alpha_{ij}$ from (5). The lemma now follows as $OPT \geq \mathsf{E}[obj(R)]$.

For any sample $S \subseteq Q_z$, define its *canonical split* as $(T_S, \overline{T_S})$ of S where $T_S = S \setminus OPT$ and $\overline{T_S} = S \setminus \overline{OPT}$. 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 $_i = \sum_{j \in S \setminus \overline{OPT}} \ _{ij}$ and so $\mathsf{E}_S[\ _i] = \sum_{j \in \overline{OPT}} \ _{ij} \ \mathbf{P}[j \in S] = p \ \ O_i$. We now have the following concentration bound.

Lemma 15. $\Pr_S[|\ _i \ \ \mathsf{E}[\ _i]| > \ Np] \ \ \frac{2}{N^3}$ *for all* $i \in Q_z$.

So, we have $\ _i \in [p \ O_i \ \ Np]$, 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( ) for the split $(T_S; \overline{T_S})$, and $LP(\hat{\mathbf{x}})$ be the corresponding value. Then, we have the following.

Lemma 16. $\mathsf{E}_S[LP(\hat{\mathbf{x}})] \ \ (1 \ \ \frac{3}{N^2})OPT \ \ N \sum_{i \in Q_z} v_i r_i$.

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

Lemma 17. $\mathsf{E}_S[ALG] \ \ (1 \ \ 8 \ ) \ OPT$.

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 objective at least $(1 \ \ 16 \ )OPT$. As the algorithm's objective is always at most OPT, its expected objective is at most $q \ \ OPT + (1 \ \ q) \ \ (1 \ \ 16 \ )OPT$. From Theorem 11, the expected objective is at least $(1 \ \ 8 \ )OPT$, which implies $q \ \ \frac{1}{2}$. By repeating the entire algorithm independently T times and picking the best solution, it follows that we obtain a solution of value at least $(1 \ \ 16 \ )OPT$ with probability at least $1 \ \ 2^{\ T}$.

Finally, to prove Theorem 10, we combine this result with the binary-search based framework from §5 (see Theorem 7). For any $\in (0;1)$, let $( ) = 1 \ \ 16$ denote the approximation ratio obtained above for the max-dicut instance (11). Setting $=$ in Theorem 7, we obtain an $( ) \ =$ $1 \ \ 17$ approximation ratio for the unconstrained PCL assortment problem. This assumes that the randomized PTAS does indeed obtain an $( )$ approximation for every max-dicut instance. Note that the number of max-dicut instances that need to be solved in the algorithm of Theorem 7 is $I = O(\log(\ _r \ _v =$). Using $T = O(\log(nI))$ repetitions of Algorithm 3 for each max-dicut instance, it follows that we obtain an $( )$-approximation for *every* instance with probability at least 1

I 2 ^{T} 1 $\frac{1}{n^2}$. Thus, for any 2 $(0;1)$ we obtain a 1 17 approximation ratio for the PCL assortment problem with probability at least 1 $o(1)$, which implies Theorem 10.

Capacitated assortment optimization: Using the above techniques, and some more ideas, we obtain the following result (see Appendix E).

Theorem 12. *There is a randomized polynomial-time approximation scheme for capacitated assortment optimization in the PCL model when the capacity c n where* > 0 *is some constant.*

7. Computational Results

In this section, we provide a summary of computational results of our approximation algorithms for unconstrained, capacitated, 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 1, 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 $(\mathbf{x}_{ALG})/\hat{z}$ where $\mathbf{x}_{ALG}$ is our solution. This gives a lower bound (percentage) on how close our solution is to the optimal.

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 $r_i = 1$ 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 $(T;n;^{-};P_0)$ 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 $[0;^{-}]$, and the value P_0 is used

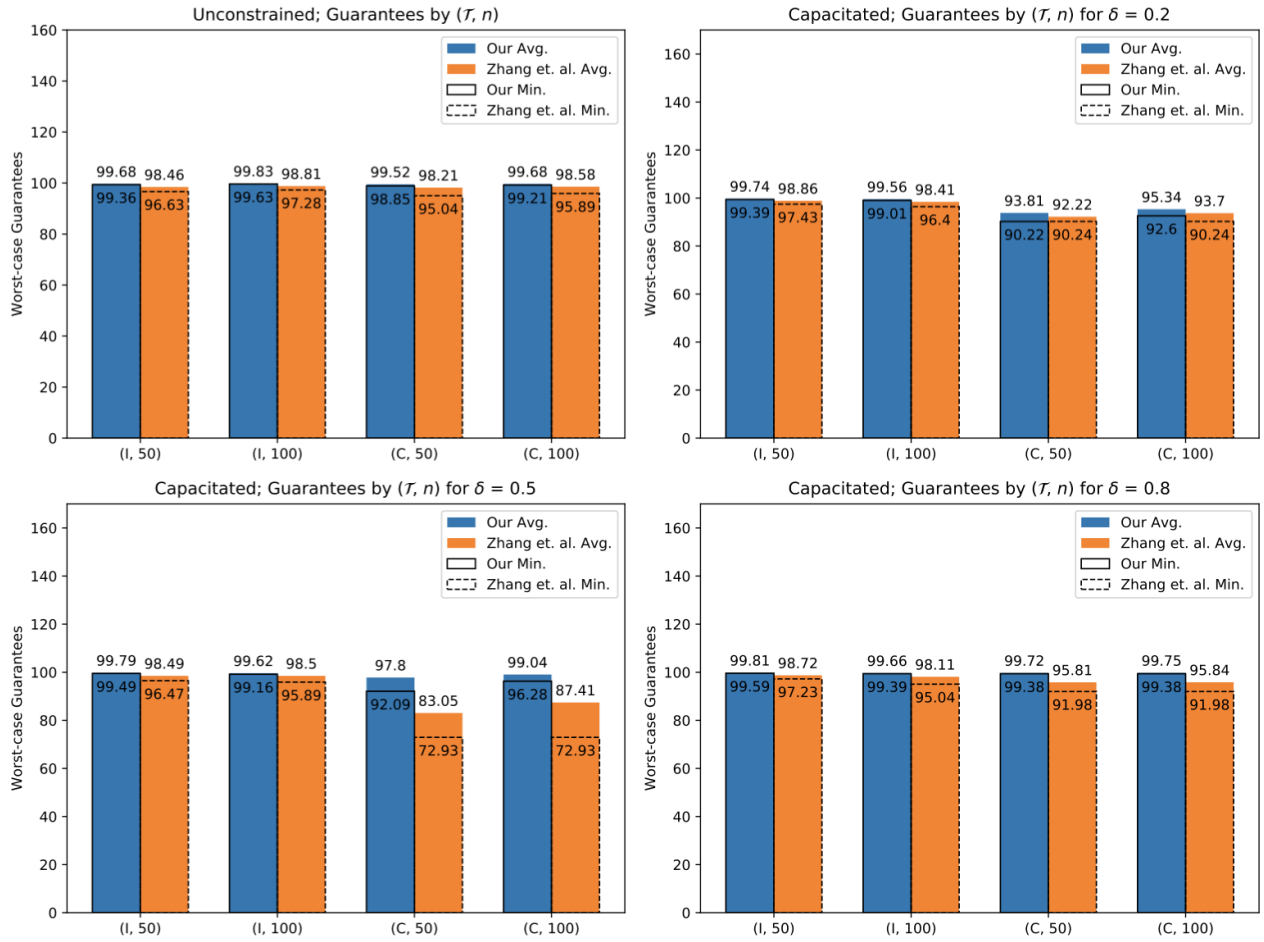


Figure 2 Comparing our algorithm and Zhang et al. (2020). The first plot (top left) is for the unconstrained case. The remaining three plots are for the capacitated case.

to generate the weight of the no-purchase option. In this way, 36 configurations are generated. For the capacitated test problems, we have an additional parameter ; we set the capacity $c = d\ ne$. These test problems are labeled as $(T; n; ^{-}; P_0;\ )$. We generate 72 configurations this way. For each configuration, 100 test instances are generated and solved using our approximation algorithm.

Reported quantities. For conciseness, we group the results by $(T; n)\ 2\ fI; Cg\quad f50; 100g$. For the unconstrained assortment problem, we obtain 4 bar plots, which are all plotted on the same graph. Each configuration of $(T; n)$ involves 9 configurations of $(^{-}; P_0)$. Over these 900 instances, we record the average performance, and the worst-case performance. We use three graphs in the capacitated case, wherein each graph corresponds to a value of $2\ f0:2; 0:5; 0:8g$. As in the uncon-

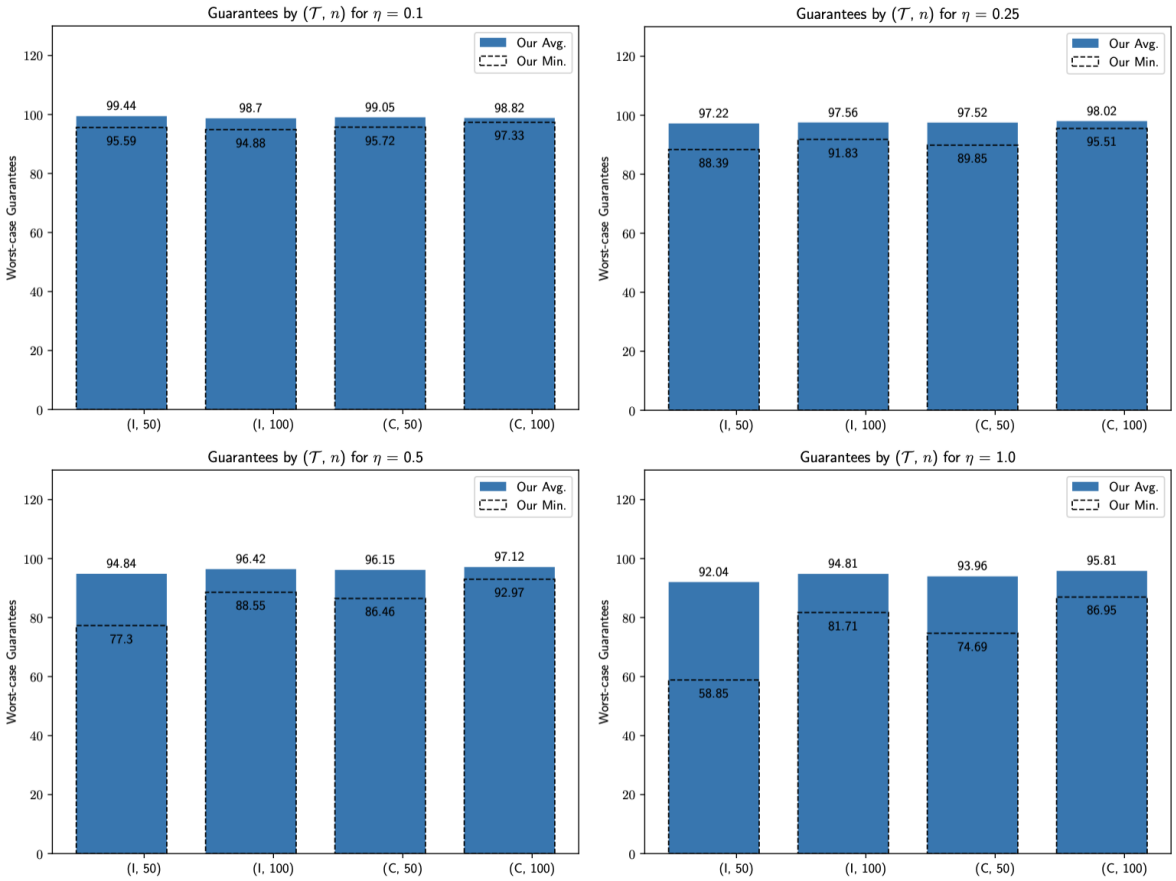


Figure 3 Summary of results for knapsack constrained assortment optimization.

strained case, the configurations are grouped by $(T; n)$ values. Additional details can be found in Appendix F

Comparison to Zhang et al. (2020). For the unconstrained and capacitated 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), while we see an improvement of about 3% in the worst-case performance. The differences are more pronounced in the capacitated instances. For = 0:2 and 0:8, the difference in average performance is similar to the unconstrained case, but the worst-case guarantees show a larger gap; e.g., we observe 8% difference in the worst-case for group $(C; 100)$ when = 0:8. On configurations with = 0:5, our algorithm outperforms Zhang et al. (2020) by over 10% on average and by almost 15% in the worst-case.

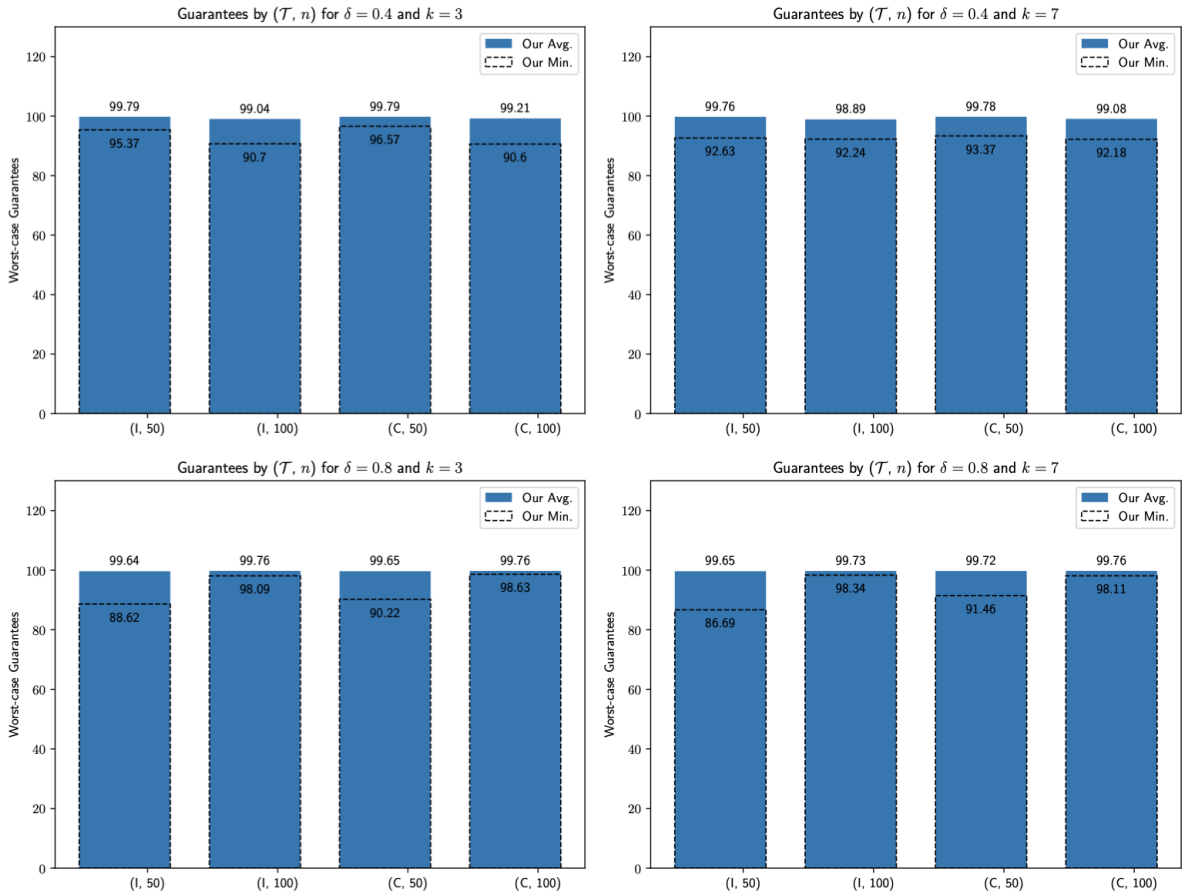


Figure 4 Summary of results for partition constrained assortment optimization.

Knapsack and partition constraints. Figures 3 and 4 summarize results for knapsack and partition constrained assortment optimization. For the knapsack case, we set the capacity to 1 and the test problems are labeled $(T; n; \bar{}; P_0; \ )$ where $\ 2 \{0; 1; 0.25; 0.5; 1\}$ and each product's size is chosen uniformly from $[0; \ ]$. Test instances for partition constraints are labeled $(T; n; \bar{}; P_0; \ ; k)$ where $k \ 2 \{3; 7\}$ is the number of parts and $\ 2 \{0.4; 0.8\}$ is the capacity constraint on each part. We generate 96 configurations for both cases, and for each configuration, we test on 100 random instances. In the knapsack case, we divide the configurations in into 4 sets based on values. We also divide the configurations in the partition constrained case into 4 sets, based on $(\ ; k)$ values. For each set, we further group the configurations according to $(T; n)$. As before, we record the average and worst-case performance guarantee in our plots.

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

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, capacitated 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 capacitated problem when the capacity is not too small. Obtaining similar improvements for the other constrained versions is an interesting open question.

Acknowledgements

We thank the three referees and AE for several helpful comments and suggestions that improved this paper. In particular, their suggestion to further investigate the structure of our 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. *IPCO*, volume 1610 of *Lecture Notes in Computer Science*, 17–30 (Springer).

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

Bekhor S, Prashker J (1998) Investigation of Stochastic Network Loading Procedures. *Transportation Research Record*, volume 1645(1), 94–102.

Berge C (1963) *Topological Spaces* (Princeton).

Blanchet J, Gallego G, Goyal V (2016) A markov chain approximation to choice modeling. *Operations Research* 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? *Computers and Operations Research*, volume 43(1), 68–77.

Davis J, Gallego G, Topaloglu H (2013) Assortment planning under the multinomial logit model with totally unimodular constraint structures. *Technical Report* .

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

Feldman J (2017) Technical Note: Space Constrained Assortment Optimization under the Paired Combinatorial Logit Model. *Available at SSRN* URL `http://dx.doi.org/https://ssrn.com/abstract=3013321`.

Feldman J, Paul A (2018) Relating the approximability of the fixed cost and space constrained assortment problems. *Production and Operations Management* URL `http://dx.doi.org/10.1111/poms.12983`.

Feldman J, Topaloglu H (2015) Capacity Constraints Across Nests in Assortment Optimization under the Nested Logit Model. *Operations Research*, volume 63(4), 812–822.

Feldman M, Naor JS, Schwartz R (2011) A unified continuous greedy algorithm for submodular maximization. *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 570–579.

Fiege U, Goemans M (1995) Approximating the Value of Two Prover Systems, With Applications to MAX 2SAT and MAX DICUT. *Proceedings of the 3rd Israel Symposium on the Theory of Computing Systems*, 182–189.

Gallego G, Topaloglu H (2014) Constrained Assortment Optimization for the Nested Logit Model. *Management Science*, volume 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.

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

Karoonsoontawong A, Lin DY (2015) Combined Gravity Model Trip Distribution and Paired Combinatorial Logit Stochastic User Equilibrium Problem. *Networks and Spatial Economics*, volume 15(4), 1011–1048.

Khot S (2002) On the power of unique 2-prover 1-round games. *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 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 Research Record B: Methodological*, volume 34(2), 65–73.

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. *9th Integer Programming and Combinatorial Optimization Conference, Cambridge, MA, USA, May 27-29, 2002, Proceedings*, 67–82.

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

McFadden D (1974) Conditional Logit Analysis of Qualitative Choice Behavior. Zarembka P, ed., *Frontiers in Economics*, 105–142.

McFadden D (1976) Quantal Choice Analaysis: A Survey. *Annals of Economic and Social Measurement*, volume 5, 363–390.

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

Rusmevichientong P, Shmoys DB, Topaloglu H (2010b) Assortment optimization with mixtures of logits. *Technical Report* .

Talluri K, van Ryzin G (2004) Revenue Management under a General Discrete Choice Model of Consumer Behavior. *Management Science*, volume 50(1), 15–33.

Zhang H, Rusmevichientong P, Topaloglu H (2020) Assortment Optimization under the Paired Combinatorial Logit Model. *Operations Research (published online)* .

Appendix A: Missing Proofs from Section 3

Proof of Lemma 1 Let $\mathbf{x} = \mathbf{1}_S$; so $x_i = 1$ if $i \in S$ and $x_i = 0$ if $i \in [n] \setminus S$. Recall that $h_z(\mathbf{x}) = \sum_{(i,j)\in M} V_{ij}(\mathbf{x})\, \gamma_{ij}(R_{ij}(\mathbf{x}) - z)$. Let $cut(S) = \sum_{(i,j)\in E: i\in S, j\notin S} w_{ij}(z)$ denote the weight of the cut S in graph G_z. We will show that the contribution of each (i,j) pair to $h_z(\mathbf{x})$ and $cut(S)$ is the same. This suffices to prove the lemma because the weight of any edge in G_z is the sum of weights added due to each pair (i,j) and $h_z(\mathbf{x})$ is clearly the sum of its contributions over (i,j) pairs. Now, consider an arbitrary ordered pair $(i,j) \in [n] \times [n]$. The contribution of (i,j) to $h_z(\mathbf{x})$ is $V_{ij}(\mathbf{x})\, \gamma_{ij}(R_{ij}(\mathbf{x}) - z)$. The contribution of (i,j) to $cut(S)$ is obtained by considering only the weights added to the four edges (i,j), (j,i), $(i,n+1)$ and $(j,n+1)$ of G_z *due to* pair (i,j); see also Figure 1(a). We consider four cases depending on x_i and x_j.

1. If both $x_i = 0$ and $x_j = 0$, the contribution to $h_z(\mathbf{x})$ is zero and the corresponding contribution to $cut(S)$ is zero.
2. If $x_i = 1$ and $x_j = 1$, then the contribution to $h_z(\mathbf{x})$ is $\frac{\overline{r}_i v_i^{1-\gamma_{ij}} + \overline{r}_j v_j^{1-\gamma_{ij}}}{(v_i^{1-\gamma_{ij}} + v_j^{1-\gamma_{ij}})^{1/\gamma_{ij}}}$. The edges cut in this case are $(i,n+1)$ and $(j,n+1)$ because $n+1 \notin S$. By definition, the weight on these edges is α^+_{ij} and α_{ji}, and we have $\alpha^+_{ij} + \alpha_{ji} = \frac{\overline{r}_i v_i^{1-\gamma_{ij}} + \overline{r}_j v_j^{1-\gamma_{ij}}}{(v_i^{1-\gamma_{ij}} + v_j^{1-\gamma_{ij}})^{1/\gamma_{ij}}}$.
3. If $x_i = 1$ and $x_j = 0$, the contribution of this pair to $h_z(\mathbf{x})$ is $v_i \overline{r}_i$. The edges crossing the cut are (i,j) and $(i,n+1)$, with weights α^+_{ij} and α^+_{ij} respectively. We have $\alpha^+_{ij} + \alpha^+_{ij} = v_i \overline{r}_i$.
4. If $x_i = 0$ and $x_j = 1$, the contribution of this pair to $h_z(\mathbf{x})$ is $v_j \overline{r}_j$. The edges crossing the cut are (j,i) and $(j,n+1)$, with weights α_{ji} and α_{ji} respectively. We have $\alpha_{ji} + \alpha_{ji} = v_j \overline{r}_j$.

Proof of Lemma 2 We prove this lemma by showing that the weight of an edge (i,j) leaving vertex i has the form $\overline{r}_i \quad _{ij}$ where $_{ij} \quad 0$ depends on the parameters v_i, v_j, γ_{ij}. First, notice that from the definitions of α^+_{ij} and $_{ij}$, we can write $\alpha^+_{ij} = \overline{r}_i \quad \alpha^+_{ij}$ for $\alpha^+_{ij} = \frac{v_i^{1-\gamma_{ij}}}{(v_i^{1-\gamma_{ij}} + v_j^{1-\gamma_{ij}})^{1/\gamma_{ij}}}$, and $_{ij} = \overline{r}_i \quad _{ij}$ for $_{ij} = \frac{v_i^{1-\gamma_{ji}}}{(v_i^{1-\gamma_{ji}} + v_j^{1-\gamma_{ji}})^{1/\gamma_{ji}}}$. Note that $0 \quad \alpha^+_{ij} \quad v_i$. To see this, consider the following chain of inequalities

$$\alpha^+_{ij} = \frac{v_i^{1-\gamma_{ij}}}{(v_i^{1-\gamma_{ij}} + v_j^{1-\gamma_{ij}})^{1/\gamma_{ij}}} \quad \frac{v_i^{1-\gamma_{ij}}}{(v_i^{1-\gamma_{ij}})^{1/\gamma_{ij}}} = \frac{v_i^{1-\gamma_{ij}}}{v_i^{(1-\gamma_{ij})\frac{1}{1}}} = v_i.$$

Similarly, $0 \quad _{ij} \quad v_i$. For any $i \in [n]$, we now consider two cases for edge $(i;j)$.

Case 1: $j = n+1$. For this case, we have $w_{ij}(z) = \sum_{k \in [n] \setminus fig} {}_{ik} = \sum_{k \in [n] \setminus fig} \overline{r}_i \; (\; {}^{+}_{ik} + \; _{ik})$. Setting $_{ij} = \sum_{k \in [n] \setminus fig} (\; {}^{+}_{ik} + \; _{ik})$ gives us $w_{ij}(z) = \overline{r}_i \quad _{ij}$, and it is easy to see that $_{ij} \quad 0$.

Case 2: $j \in [n]$. For such edges, $w_{ij}(z) = \; _{ij} = \; {}^{+}_{ij} + \; _{ij} = 2v_i\overline{r}_i \quad {}^{+}_{ij} \quad _{ij} = \overline{r}_i \quad _{ij}$ for $_{ij} = 2v_i \quad {}^{+}_{ij} \quad _{ij}$. It is again easy to see that $_{ij} \quad 0$.

Thus, for any edge $(i;j)$, we can write $w_{ij}(z) = \overline{r}_i \quad _{ij}$ for $_{ij} \quad 0$. This proves the lemma.

Proof of Lemma 3 Given the optimal solution $\mathbf{x}$, construct $\overline{\mathbf{x}}$ as follows:

$$\overline{x}_i = \begin{cases} 0; & \text{if } i \in N_z \\ x_i; & \text{if } i \in P_z = V \setminus N_z \end{cases}$$

By the downward closure of the feasible set F, solution $\overline{\mathbf{x}}$ is also feasible. We claim that the objective value of $\overline{\mathbf{x}}$ is at least as much as the objective value of $\mathbf{x}$. The contribution of a pair $(i;j)$ to the objective value is

$$\frac{\overline{r}_i v_i^{1= \; _{ij}} x_i + \overline{r}_j v_j^{1= \; _{ij}} x_j}{V_{ij}(\mathbf{x}\;)^{1 \quad _{ij}}}$$

Notice that if $i;j \in P_z$, the contribution does not change. Suppose that exactly one of $i;j$ is in N_z. Without loss of generality, let $i \in N_z$. Then, we have

$$\overline{r}_i v_i^{1= \; _{ij}} x_i + \overline{r}_j v_j^{1= \; _{ij}} x_j \qquad \overline{r}_i v_i^{1= \; _{ij}} \overline{x}_i + \overline{r}_j v_j^{1= \; _{ij}} \overline{x}_j$$

since $\overline{x}_i = 0$ and $\overline{r}_i < 0$. Moreover, the denominator $V_{ij}(\mathbf{x}\;)^{1 \quad _{ij}} \quad V_{ij}(\overline{\mathbf{x}})^{1 \quad _{ij}}$ which shows that for such a pair, its contribution to the objective value can only increase. Lastly, if both $i;j \in N_z$, a similar argument shows that the contribution of pair $i;j$ to the objective value can also only increase. Note that we follow the convention that $\frac{0}{0} = 0$. Thus, the objective value of $\overline{\mathbf{x}}$ is at least as much as the objective value of $\mathbf{x}$.

Proof of Lemma 4 Given the $\mathrm{IP}_2(z)$ solution $(\mathbf{x}\;;\mathbf{y}\;)$, construct $(\overline{\mathbf{x}};\overline{\mathbf{y}})$ as follows:

$$\overline{x}_i = \begin{cases} 0; & \text{if } i \in N_z \\ x_i; & \text{if } i \in P_z = V \setminus N_z \end{cases} \quad \text{and} \quad \overline{y}_{ij} = \min(\overline{x}_i; 1 \quad \overline{x}_j) \text{ for all } (i;j) \in E$$

We claim that $(\overline{\mathbf{x}};\overline{\mathbf{y}})$ is a feasible solution and that its objective value is at least as much as the objective value of $(\mathbf{x}\;;\mathbf{y}\;)$. Since we only decrease values of some vertices in $\mathbf{x}$ to obtain $\overline{\mathbf{x}}$, the $\overline{\mathbf{x}}$ vector is feasible by the downward closure of F. Moreover we changed $\overline{\mathbf{y}}$ accordingly so that $(\overline{\mathbf{x}};\overline{\mathbf{y}})$ is a feasible solution for $\mathrm{IP}_2(z)$.

Next, we show that the objective value of the new solution does not decrease. The change in the objective value is

$$\sum_{(i;j) \in E} \overline{y}_{ij} w_{ij}(z) \qquad \sum_{(i;j) \in E} y_{ij} w_{ij}(z)$$

which can be rewritten as

$$\sum_{(i;j)\in E:i\in N_z} (\bar{y}_{ij} - y_{ij})\, w_{ij}(z) + \sum_{(i;j)\in E:i\in P_z} (\bar{y}_{ij} - y_{ij})\, w_{ij}(z).$$

Consider an edge $(i;j) \in E$. If $i \in N_z$, $\bar{x}_i = 0$, and $\bar{y}_{ij} = 0$. From Lemma 2, we know $w_{ij}(z) < 0$. Since $y_{ij} \geq 0$, we can conclude that $\sum_{(i;j)\in E:i\in N_z}(\bar{y}_{ij} - y_{ij})\, w_{ij}(z) \geq 0$. If, on the other hand, $i \in P_z$, then $\bar{x}_i = x_i$, and $1 - \bar{x}_j \geq 1 - x_j$ (either x_j stays the same or decreases). Thus $\bar{y}_{ij} \leq y_{ij}$. From Lemma 2, we know $w_{ij}(z) \geq 0$, and we have $\sum_{(i;j)\in E:i\in P_z}(\bar{y}_{ij} - y_{ij})\, w_{ij}(z) \geq 0$. It thus follows that

$$\sum_{(i;j)\in E} \bar{y}_{ij} w_{ij}(z) \geq \sum_{(i;j)\in E} y_{ij} w_{ij}(z) \geq 0$$

as desired.

Proof of Lemma 5. To see the statement for $\mathrm{IP}_1(z)$, note that

$$h_z(\mathbf{x}) = \sum_{(i;j)\in M} V_{ij}(\mathbf{x})^{\,ij}(R_{ij}(\mathbf{x}) - z) = \sum_{(i;j)\in M} V_{ij}(\mathbf{x})^{\,ij}\left(\frac{r_i v_i^{1=\,ij} x_i + r_j v_j^{1=\,ij} x_j - z(v_i^{1=\,ij} x_i + v_j^{1=\,ij} x_j)}{V_{ij}(\mathbf{x})}\right)^{!}$$
$$= \sum_{(i;j)\in M} V_{ij}(\mathbf{x})^{\,ij}\left(\frac{(r_i - z)v_i^{1=\,ij} x_i + (r_j - z)v_j^{1=\,ij} x_j}{V_{ij}(\mathbf{x})}\right)^{!} = \sum_{(i;j)\in M} \frac{\overline{r}_i v_i^{1=\,ij} x_i + \overline{r}_j v_j^{1=\,ij} x_j}{V_{ij}(\mathbf{x})^{1-\,ij}}.$$

So $\mathrm{IP}_1(z)$ is just $\max_{\mathbf{x}\in F} h_z(\mathbf{x}) = f(z)$.

We now consider the second statement. It is clear that the optimal value of $\mathrm{IP}_2(z)$ is at least the optimal max-dicut value on G_z subject to the constraint F: for any $S \subseteq [n]$ consider the $\mathrm{IP}_2(z)$ solution $\mathbf{x} = \mathbf{1}_S$ and $y_{ij} = x_i(1 - x_j)$, which has objective equal to the cut value of S. We now show that the optimal value of max-dicut on G_z subject to constraint F is at least that of $\mathrm{IP}_2(z)$. By Lemma 4, there is an optimal solution $(\bar{\mathbf{x}};\bar{\mathbf{y}})$ for $\mathrm{IP}_2(z)$ with $\bar{x}_i = 0$ for all $i \in N_z$. From the constraints in $\mathrm{IP}_2(z)$, we have $\bar{y}_{ij} \leq \min\{\bar{x}_i; 1 - \bar{x}_j\} = \bar{x}_i(1 - \bar{x}_j)$. We claim that $\bar{y}_{ij} = \bar{x}_i(1 - \bar{x}_j)$ for all $i;j$. Indeed,

If $\bar{x}_i = 0$ or $\bar{x}_j = 1$ then clearly $\bar{y}_{ij} = 0$.

If $\bar{x}_i = 1$ then it must be that $i \in P_z$. By Lemma 2, all edges out of i have non-negative weight.

So $w_{ij}(z) \geq 0$. By optimality of $(\bar{\mathbf{x}};\bar{\mathbf{y}})$ it follows that $\bar{y}_{ij} = \bar{x}_i(1 - \bar{x}_j)$.

So the optimal value of $\mathrm{IP}_2(z)$ is $\sum_{(i;j)\in E} w_{ij}(z)\, \bar{x}_i(1 - \bar{x}_j)$, which is the directed cut value of $\{i \in V : \bar{x}_i = 1\}$ in G_z. So the optimal value of max-dicut on G_z (under constraint F) is at least that of $\mathrm{IP}_2(z)$.

Proof of Lemma 6 We first prove the equivalence of $\mathrm{IP}_1(z)$ and $\mathrm{IP}_2(z)$. By the first statement in Lemma 5, the optimal value of $\mathrm{IP}_1(z)$ is $f(z)$. Let $S \subseteq [n]$ denote the optimal solution, i.e., $f(z) = h_z(\mathbf{1}_S)$. By the second statement in Lemma 5, the optimal value of $\mathrm{IP}_2(z)$ is the same as

that of the max-dicut instance (under constraint F). Let $S^0 \subseteq [n]$ denote this optimal solution, i.e., the optimal value of $\mathrm{IP}_2(z)$ is $cut(S^0)$ the directed cut value of S^0. By Lemma 1, we have $h_z(\mathbf{1}_S) = cut(S)$ and $h_z(\mathbf{1}_{S^0}) = cut(S^0)$. We now have:

$$cut(S) = h_z(\mathbf{1}_S) \quad h_z(\mathbf{1}_{S^0}) = cut(S^0) \quad cut(S);$$

where the first inequality is because S is the optimal solution to $f(z)$ and the second inequality is because S^0 is the optimal solution to the max-dicut instance. Thus, we have equality throughout, which proves the equivalence of $\mathrm{IP}_1(z)$ and $\mathrm{IP}_2(z)$.

We now show that the optimal value of $\mathrm{IP}_2(z)$ equals the optimal max-dicut value on $\mathcal{G}_z$. By Lemma 5, the optimal $\mathrm{IP}_2(z)$ value equals the optimal max-dicut value on G_z. We will show that the optimal max-dicut value on $\mathcal{G}_z$ (OPT) equals that on G_z (OPT). To see this, for any subset $S \subseteq V$, let $cut(S)$ and $\widehat{cut}(S)$ denote the values of the cut S in G_z and $\mathcal{G}_z$ respectively. Let $E_P(S)$ (resp. $E_N(S)$) denote the total weight of edges from $S \setminus P_z$ to $V \setminus S$ (resp. $S \setminus N_z$ to $V \setminus S$). So $cut(S) = E_P(S) + E_N(S)$ and $\widehat{cut}(S) = E_P(S)$. By definition of $\mathcal{G}_z$ and Lemma 2, we have $E_N(S) \geq 0$ and so $\widehat{cut}(S) \geq cut(S)$ for all $S \subseteq V$, which implies $OPT \geq OPT$. For the other direction, let $\tilde{S}$ denote an optimal solution on graph $\mathcal{G}_z$. We may assume that $\tilde{S} \supseteq P_z$ as $\mathcal{G}_z$ does not have any out-going edges from vertices not in P_z. So $cut(\tilde{S}) = \widehat{cut}(\tilde{S})$, which implies $OPT \leq OPT$.

A.1. Complexity of the Max-Dicut Reduction

Even though the input to the assortment problem is rational, the weights $w_{ij}(z)$ in our reduction may be irrational. As we saw in the proof of Lemma 2, we can write $w_{ij}(z) = (r_i \quad z) \quad _{ij}$ where $0 \quad _{ij} \quad 2nv_{max}$. We now give a detailed approach to deal with irrational $_{ij}$ values. We define $:= \frac{1}{r_{\max} n^3}$, and for every $_{ij}$, pick a rational $\overline{\ _{ij}}$ such that $j \ _{ij} \quad \overline{\ _{ij}} j \quad$. Dirichlet's approximation theorem guarantees the existence of $\overline{\ _{ij}}$. where the denominator is at most $\frac{1}{}$ (so the numerator is at most $\frac{2nv_{max}}{}$). Note that the bit complexity of each rational $\overline{\ _{ij}}$ is $O(\log(n\, r_{max}\, v_{max}))$ which is polynomial. The method of continued fractions can be used to compute $\overline{\ _{ij}}$ in polynomial time. With this, we define a new integer program as follows.

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

where $\overline{w_{ij}}(z) = \overline{r_i} \quad \overline{_{ij}}$ for $\overline{_{ij}}$ as defined earlier. The integer programs $\mathrm{IP}_2(z)$ and $\overline{\mathrm{IP}}_2(z)$, which have identical feasible regions, satisfy the following lemma.

Lemma 18. *Fix any z. Let $(\mathbf{x};\mathbf{y})$ be a feasible solution for $IP_2(z)$ and $\overline{IP}_2(z)$. Let $(\mathbf{x};\mathbf{y})$ and $\overline{\ }(\mathbf{x};\mathbf{y})$ denote the objective value of $(\mathbf{x};\mathbf{y})$ for $IP_2(z)$ and $\overline{IP}_2(z)$ respectively. Then*

$$j \quad (\mathbf{x};\mathbf{y}) \quad \overline{\ }(\mathbf{x};\mathbf{y})j \quad \frac{1}{n}:$$

Moreover,

$$j \max_{\mathbf{x};\mathbf{y}} \quad (\mathbf{x};\mathbf{y}) \quad \max_{\mathbf{x};\mathbf{y}} \overline{\ }(\mathbf{x};\mathbf{y})j \quad \frac{1}{n}:$$

In words, the di erence in the optimal values of the formulations is bounded by $O(\frac{1}{n})$.

Proof of Lemma 18 For the first part, we have

$$j \quad (\mathbf{x};\mathbf{y}) \quad \overline{\ }(\mathbf{x};\mathbf{y})j = j \sum_{(i;j)2E} w_{ij}(z) \quad y_{ij} \quad \sum_{(i;j)2E} \overline{w_{ij}}(z) \quad y_{ij}j = j \sum_{(i;j)2E} r_i(\quad _{ij} \quad \overline{_{ij}})y_{ij}j$$
$$n^2 \quad r_{\max} \quad = \frac{1}{n};$$

where the inequality follows from the fact that $jEj \quad n^2$, $z \quad 0$, so $\overline{r_i} = r_i \quad z \quad r_{\max}$, and the fact that $j \quad _{ij} \quad \overline{_{ij}}j \quad$. For the second part, let $(\mathbf{x}^0;\mathbf{y}^0)$ and $(\mathbf{x}^{00};\mathbf{y}^{00})$ be the optimal solutions to $\mathrm{IP}_2(z)$ and $\overline{\mathrm{IP}}_2(z)$ respectively. We claim that $j \quad (\mathbf{x}^0;\mathbf{y}^0) \quad \overline{\ }(\mathbf{x}^{00};\mathbf{y}^{00})j \quad \frac{1}{n}$. We have

$$(\mathbf{x}^0;\mathbf{y}^0) \quad \overline{\ }(\mathbf{x}^{00};\mathbf{y}^{00}) \qquad (\mathbf{x}^0;\mathbf{y}^0) \quad \overline{\ }(\mathbf{x}^0;\mathbf{y}^0) \quad \frac{1}{n}$$

where the first inequality is due to the fact that $\overline{\ }(\mathbf{x}^{00};\mathbf{y}^{00}) \quad \overline{\ }(\mathbf{x}^0;\mathbf{y}^0)$, and the second inequality follows from the first part of the lemma. Furthermore, using the fact that $(\mathbf{x}^0;\mathbf{y}^0) \qquad (\mathbf{x}^{00};\mathbf{y}^{00})$ and the first part of the lemma, we get

$$(\mathbf{x}^0;\mathbf{y}^0) \quad \overline{\ }(\mathbf{x}^{00};\mathbf{y}^{00}) \qquad (\mathbf{x}^{00};\mathbf{y}^{00}) \quad \overline{\ }(\mathbf{x}^{00};\mathbf{y}^{00}) \qquad \frac{1}{n}:$$

This proves the desired result.

As a consequence of Lemma 18, at the loss of a $1 + o(1)$ factor in the objective, we can assume that all edge weights in the reduction to max-dicut are rational with polynomial bit complexity.

Appendix B: Algorithm for unconstrained assortment optimization

We define the upper-bound $g(z)$ as the optimal value of the following SDP relaxation. This was used to obtain a 0:874-approximation algorithm for max-dicut with non-negative weights by Lewin et al. (2002).

$$g(z) = \text{maximize: } \frac{1}{4} \sum_{(i;j)2E} w_{ij}(z)(1 + \mathbf{u_0} \quad \mathbf{u_i} \quad \mathbf{u_0} \quad \mathbf{u_j} \quad \mathbf{u_i} \quad \mathbf{u_j}) \qquad \text{(SDP}(z)\text{)}$$

$$\text{subject to: } \mathbf{u_0}\ \mathbf{u_i} + \mathbf{u_0}\ \mathbf{u_j} + \mathbf{u_i}\ \mathbf{u_j} \quad 1 \qquad 1 \quad i;j \quad n+1 \tag{13}$$

$$\mathbf{u_0}\ \mathbf{u_i} \quad \mathbf{u_0}\ \mathbf{u_j} + \mathbf{u_i}\ \mathbf{u_j} \quad 1 \qquad 1 \quad i;j \quad n+1 \tag{14}$$

$$\mathbf{u_0}\ \mathbf{u_i} + \mathbf{u_0}\ \mathbf{u_j} \quad \mathbf{u_i}\ \mathbf{u_j} \quad 1 \qquad 1 \quad i;j \quad n+1 \tag{15}$$

$$\mathbf{u_0}\ \mathbf{u_i} \quad \mathbf{u_0}\ \mathbf{u_j} \quad \mathbf{u_i}\ \mathbf{u_j} \quad 1 \qquad 1 \quad i;j \quad n+1 \tag{16}$$

$$\mathbf{u_{n+1}} = \mathbf{u_0} \tag{17}$$

$$\mathbf{u_i} \ 2 \ \mathsf{R}^{n+2};\ \mathbf{u_i}\ \mathbf{u_i} = 1 \qquad 0 \quad i \quad n+1 \tag{18}$$

where every $\mathbf{u_i}$ for $i \ 2 \ V$ is a vector corresponding to vertex i. At a high level, the vector $\mathbf{u_0}$ is used to determine which vertices are placed in the cut. To see that SDP(z) is a relaxation, note that for any cut $S \quad V$, we can set $\mathbf{u_i} = (1;0;:::;0) \ 2 \ \mathsf{R}^{n+2}$ for $i \ 2 \ S$ and $i = 0$, and $\mathbf{u_i} = (\ \ 1;0;:::;0) \ 2 \ \mathsf{R}^{n+2}$ otherwise. This is a feasible solution, and the objective value of this solution is equal to the value of the cut $(S;V \ n \ S)$. From Lemma 6 it follows that $g(z)$ is a valid upper bound for $f(z)$.

Since G_z may have edges of negative weight, we cannot directly use the approximation guarantee from Lewin et al. (2002). We prove a lemma analogous to Lemma 7 that allows us to drop all edges with negative weight in G_z (without loss of generality).

Lemma 19. *Given an optimal solution $f\mathbf{u_i}g_{i=0}^{n+1}$ to SDP(z), we can construct another optimal solution $f\overline{\mathbf{u}}_\mathbf{i}g_{i=0}^{n+1}$ such that for all $i \ 2 \ N_z$, $\overline{\mathbf{u}}_\mathbf{i} = \mathbf{u_0}$, i.e. no vertex that has outgoing edges of negative weight contributes to the solution, and for all $i \ 2 \ P_z$, $\overline{\mathbf{u}}_\mathbf{i} = \mathbf{u_i}$.*

Proof. Given the SDP solution $f\mathbf{u_i}g_{i=0}^{n+1}$ construct $f\overline{\mathbf{u}}_\mathbf{i}g_{i=0}^{n+1}$ as follows:

$$\overline{\mathbf{u}}_\mathbf{i} = \begin{cases} \mathbf{u_0}; & \text{if } i \ 2 \ N_z \\ \mathbf{u_i}; & \text{if } i \ 2 \ P_z = V \ n \ N_z \text{ or } i = 0 \end{cases}$$

Notice that $\overline{\mathbf{u}}_\mathbf{0} = \mathbf{u_0}$. We claim that $f\overline{\mathbf{u}}_\mathbf{i}g_{i=0}^{n+1}$ is a feasible solution for SDP(z) and that its objective value is at least as much as the objective value of $f\mathbf{u_i}g_{i=0}^{n+1}$.

Feasibility. Clearly the new solution satisfies constraints (17) and (18). Using the fact that the vectors $f\mathbf{u_i}g_{i=0}^{n+1}$ in SDP(z) are all unit-vectors, we can rewrite constraints (13)-(16) as the following:

$$(\mathbf{u_0} + \mathbf{u_i})\ (\mathbf{u_0} + \mathbf{u_j}) \quad 0 \tag{19}$$

$$(\mathbf{u_0} \quad \mathbf{u_i})\ (\mathbf{u_0} \quad \mathbf{u_j}) \quad 0 \tag{20}$$

$$(\mathbf{u_0} \quad \mathbf{u_i})\ (\mathbf{u_0} + \mathbf{u_j}) \quad 0 \tag{21}$$

$$(\mathbf{u_0} + \mathbf{u_i})\ (\mathbf{u_0} \quad \mathbf{u_j}) \quad 0 \tag{22}$$

To see the validity of these constraints, consider the following cases:

1. $i;j \ 2 \ P_z$. Then all constraints are satisfied due to the feasibility of $f\mathbf{u_i}g_{i=0}^{n+1}$.

2. At least one of $i;j$ (say i) is in N_z, so $\bar{\mathbf{u}}_\mathbf{i} = \bar{\mathbf{u}}_\mathbf{0}$. The left-hand-side in constraints (19) and (22) are zero, so they are trivially satisfied. The left-hand-side in constraint (20) is $2(\bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{j}) = 2(1 \ \bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{j})$. Similarly, the left-hand-side in constraint (21) is $2(1 + \bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{j})$. As $\bar{\mathbf{u}}_\mathbf{0}$ and $\bar{\mathbf{u}}_\mathbf{j}$ are unit vectors, we have $|\bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{j}| \ 1$ which implies the validity of (20) and (21).

Optimality. Note that the (SDP(z)) objective can be written as

$$\frac{1}{4} \sum_{(i;j) 2 E} w_{ij}(z) (\mathbf{u}_\mathbf{0} + \mathbf{u}_\mathbf{i}) \ (\mathbf{u}_\mathbf{0} \ \mathbf{u}_\mathbf{j}):$$

Define $o_{ij} = (\mathbf{u}_\mathbf{0} + \mathbf{u}_\mathbf{i}) \ (\mathbf{u}_\mathbf{0} \ \mathbf{u}_\mathbf{j})$ corresponding to the $(i;j)$ term in the objective for $\{\mathbf{u}_\mathbf{i}\}_{i=0}^{n+1}$. Similarly, define $\bar{o}_{ij} = (\bar{\mathbf{u}}_\mathbf{0} + \bar{\mathbf{u}}_\mathbf{i}) \ (\bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{j})$. We will show that $w_{ij}(z)(\bar{o}_{ij} \ o_{ij}) \ 0$ for each $(i;j) 2 E$, which suffices to prove the optimality of $\{\bar{\mathbf{u}}_\mathbf{i}\}_{i=0}^{n+1}$. To see this, consider the following cases:

1. $i 2 N_z$, so $\bar{\mathbf{u}}_\mathbf{i} = \bar{\mathbf{u}}_\mathbf{0}$. Here, $\bar{o}_{ij} = 0$. Moreover, $o_{ij} \ 0$ by constraint (22), and so $\bar{o}_{ij} \ o_{ij} \ 0$. Using $i 2 N_z$ and Lemma 2, we have $w_{ij}(z) < 0$. Hence $w_{ij}(z)(\bar{o}_{ij} \ o_{ij}) \ 0$.
2. $i 2 P_z$ and $j 2 N_z$, so $\bar{\mathbf{u}}_\mathbf{j} = \bar{\mathbf{u}}_\mathbf{0}$. Here, $\bar{o}_{ij} = 2(1 + \bar{\mathbf{u}}_\mathbf{0} \ \bar{\mathbf{u}}_\mathbf{i}) = 2(1 + \mathbf{u}_\mathbf{0} \ \mathbf{u}_\mathbf{i})$. So

$$\bar{o}_{ij} \ o_{ij} = 1 + \mathbf{u}_\mathbf{0} \ \mathbf{u}_\mathbf{i} + \mathbf{u}_\mathbf{0} \ \mathbf{u}_\mathbf{j} + \mathbf{u}_\mathbf{i} \ \mathbf{u}_\mathbf{j} \ 0;$$

   where the inequality is by (19). Using $i 2 P_z$ and Lemma 2, we have $w_{ij}(z) \ 0$ and hence $w_{ij}(z)(\bar{o}_{ij} \ o_{ij}) \ 0$.
3. $i;j 2 P_z$. In this case $\bar{o}_{ij} = o_{ij}$, which implies $w_{ij}(z)(\bar{o}_{ij} \ o_{ij}) = 0$.

As $g(z)$ is a valid upper bound to $f(z)$, we have $g(z) \ 0$ for all $z 2 \mathsf{R}$. To prove the existence of a fixed point for $g(\ ) = v_0$, we use the following theorem from Berge (1963).

Theorem 13. *Let Z and U be topological spaces, $: Z \ U \ ! \ \mathsf{R}$ be a continuous function, and $C : Z \ ! \ 2^U$ be a compact valued correspondence such that $C(z) \neq ;$ for all $z 2 Z$. Define $: Z \ ! \ \mathsf{R}$ by $(z) = \sup\{ (z; \ ) \ j \ 2 C(z)\}$. If C is continuous at z, then is also continuous at z.*

Lemma 20. *There exists a fixed point for $g(\ ) = v_0$.*

Proof. Let $\{\mathbf{u}_\mathbf{i}\}_{i=0}^{n+1}$ denote any solution to SDP(z), and let $U \ \mathsf{R}^{(n+2) \ (n+2)}$ be the space of feasible solutions to SDP(z). Note that U is compact. Moreover, let $(z;u) = \frac{1}{4} \sum_{(i;j) 2 E} w_{ij}(z)(1 + \mathbf{u}_\mathbf{0} \ \mathbf{u}_\mathbf{i} \ \mathbf{u}_\mathbf{0} \ \mathbf{u}_\mathbf{j} \ \mathbf{u}_\mathbf{i} \ \mathbf{u}_\mathbf{j})$ which is quadratic in the variables $\{\mathbf{u}_\mathbf{i}\}_{i=0}^{n+1}$, and linear in z. Hence $(z;u)$ is continuous in all variables. Let the space $Z = \mathsf{R}$. Note that (z) is the same as $g(z)$ for all $z 2 \mathsf{R}$. For every $z 2 Z$, the space of solutions U to SDP(z) does not change, and thus the correspondence C is constant, and hence continuous. Moreover, U is compact, and so C is a compact-valued correspondence for all $z 2 Z$. We have set-up the above notation in correspondence with Theorem 13. Thus, we can conclude that $g(z)$ is continuous for all $z 2 \mathsf{R}$.

For any $(i;j) \in E$, note that $w_{ij}(z)$ has a negative linear dependence on z and by (16) we have $(1 + \mathbf{u_0}\ \mathbf{u_i}\ \ \mathbf{u_0}\ \mathbf{u_j}\ \ \mathbf{u_i}\ \mathbf{u_j}) \quad 0$. So $g(z)$ is a non-increasing function of z.

On the other hand, $v_0 z$ is increasing in z. Note that at $z = 0$, $g(z) \quad 0$ and $v_0 z = 0$. As $z \ ! \ 1$, $v_0 z \ ! \ 1$. Combining these observations, we conclude that there is $\hat{z} \quad 0$ with $g(\hat{z}) = v_0 \hat{z}$.

We now reformulate SDP(z) based on the following observations and notations. Given constraints (17) and (18), all the constraints (13), (14), (15), (16) when $i = n+1$ or $j = n+1$ become redundant. So, we can remove all constraints involving variable $\mathbf{u_{n+1}}$ and replace all occurrences of $\mathbf{u_{n+1}}$ in the objective by $\mathbf{u_0}$. Note that this does not change the objective value $g(z)$ of SDP(z).

The constraints (13), (14), (15), (16) are also redundant when $i = j$. After removing the variable $\mathbf{u_{n+1}}$, we switch to the "matrix view" of SDP(z). We consider the variables to be entries in a symmetric matrix $X = UU^t \in \mathbb{R}^{(n+1)\ (n+1)}$. Here each row $\mathbf{u_i}$ of matrix U corresponds to variable $\mathbf{u_i}$ of SDP(z): so the variables in the reformulation are $x_{ij} = \mathbf{u_i}\ \mathbf{u_j}$. We note that Lemma 20 still holds for this reformulation (and the proof is identical).

We define symmetric matrices $\hat{T}_{ij} \in \mathbb{R}^{(n+1)\ (n+1)}$ for $(i;j) \in E$, $j \neq n+1$ as follows.

$$\hat{T}_{ij}(\ ;\ ) = \begin{cases} \geq \frac{1}{2} & \text{if } (\ ;\ ) = (0;i) \text{ or } (i;0) \\ \frac{1}{2} & \text{if } (\ ;\ ) \in \{(0;j);(j;0);(i;j);(j;i)\}; \quad 0 \quad ; \quad n \\ > 0 & \text{otherwise} \end{cases}$$

and symmetric matrices $\overline{T}_{ij} \in \mathbb{R}^{(n+1)\ (n+1)}$ for $(i;j) \in E$, $j = n+1$ as follows.

$$\overline{T}_{ij}(\ ;\ ) = \begin{cases} 1 & \text{if } (\ ;\ ) = (0;i) \text{ or } (i;0) \\ 0 & \text{otherwise} \end{cases}; \quad 0 \quad ; \quad n.$$

Let $\langle T; X \rangle = \text{trace}(T^t X)$. Note that the objective of SDP(z) now becomes

$$\frac{1}{4} \sum_{(i;j)\in E;\, j \neq n+1} w_{ij}(z) \langle \hat{T}_{ij}; X \rangle + \frac{1}{4} \sum_{(i;j)\in E;\, j = n+1} w_{ij}(z) \langle \overline{T}_{ij}; X \rangle + \frac{1}{4} \sum_{(i;j)\in E} w_{ij}(z) + \frac{1}{4} \sum_{(i;j)\in E;\, j = n+1} w_{ij}(z)$$

We further simplify the objective by setting $T(z) = \frac{1}{4} \sum_{(i;j)\in E;\, j \neq n+1} w_{ij}(z) \hat{T}_{ij} + \sum_{(i;j)\in E;\, j = n+1} w_{ij}(z) \overline{T}_{ij}$. Notice that $T(z)$ is a symmetric matrix. We also define $c(z) = \frac{1}{4} \sum_{(i;j)\in E;\, j \neq n+1} w_{ij}(z) + \frac{1}{2} \sum_{(i;j)\in E;\, j = n+1} w_{ij}(z)$ We can now reformulate SDP(z) as follows.

$$g(z) = \text{maximize: } \langle T(z); X \rangle + c(z) \qquad (\text{SDP}_2(z))$$

$$\text{subject to: } X_{0;i} + X_{0;j} + X_{i;j} \quad 1 \qquad 1 \quad i;j \quad n;\ i \neq j \qquad (23)$$

$$X_{0;i} \quad X_{0;j} + X_{i;j} \quad 1 \qquad 1 \quad i;j \quad n;\ i \neq j \qquad (24)$$

$$X_{0;i} + X_{0;j} \quad X_{i;j} \quad 1 \qquad 1 \quad i;j \quad n;\ i \neq j \qquad (25)$$

$$X_{0;i} \quad X_{0;j} \quad X_{i;j} \quad 1 \qquad 1 \quad i;j \quad n;\ i \neq j \qquad (26)$$

$$X_{i;i} = 1 \qquad 0 \quad i \quad n$$

$$X \quad 0$$

We now transform $\mathrm{SDP}_2(z)$ into its standard form. Let K index all constraints in (23) - (26). First, we introduce a non-negative slack variable $\_k$ for each constraint $k \in K$ to convert the inequalities into equalities. For example, for a constraint k of type (23), we get $X_{0;i} + X_{0;j} + X_{i;j} \quad \_k = \quad 1$. Let Γ be a diagonal matrix of the $\_k$ slack variables. Let $Y = \begin{smallmatrix} X & 0 \\ 0 & \Gamma \end{smallmatrix}$ be a block matrix of the decision variables. Note that $Y \quad 0$ if, and only if, $X \quad 0$ and $\_k \quad 0$ for all $k \in K$. Moreover, we introduce symmetric matrices A_k for all constraints $k \in K$ to write the constraints in standard form. For $i = 0; 1; :::; n$, let $D_i = \begin{smallmatrix} e_i e_i^t & 0 \\ 0 & 0 \end{smallmatrix}$ where e_i is a unit vector with a 1 in the i^{th} position. We also modify $T(z)$ accordingly (by appending zeros corresponding to the variables in Γ).

Also, notice that for a fixed z, the term $c(z)$ in the objective function of $\mathrm{SDP}_2(z)$ is constant, and thus can be omitted. Then, we can write $\mathrm{SDP}_2(z)$ in standard form as follows. We let $g^0(z)$ denote the objective value of the new SDP. Note that $g^0(z) + c(z) = g(z)$.

$$g^0(z) = \text{maximize: } \langle T(z); Y \rangle \qquad (\mathrm{SDP}_3(z))$$

$$\text{subject to: } \langle A_k; Y \rangle = \quad 1; \ \forall k \in K \qquad (27)$$

$$\langle D_i; Y \rangle = 1; \ i = 0; 1; :::; n \qquad (28)$$

$$Y \quad 0$$

It is easy to verify that $Y = I$, a positive definite matrix, is a feasible solution to $\mathrm{SDP}_3(z)$. The dual of $(\mathrm{SDP}_3(z))$ has the following form.

$$g^0(z) = \text{minimize: } \sum_{k \in K} \_k \quad \sum_{i=0} \_i \qquad (\mathrm{SDP}_D(z))$$

$$\text{subject to: } \sum_{k \in K} A_k \ \_k + \sum_{i=0} D_i \ \_i \qquad T(z)$$

where $\_k$ are the dual variables corresponding to constraint (27), and $\_i$ are the dual variables corresponding to constraint (28). Let $\quad$ be the maximum absolute value over all eigenvalues of matrix $T(z)$. Then the following is a feasible solution for $\mathrm{SDP}_D(z)$.

$$\_i = 0 \qquad \text{for all } i \in K$$

$$\_i = \quad (1 + \ ) \qquad \text{for } i = 1; :::; n$$

Moreover, it is a strictly feasible solution, i.e. $\sum_{k \in K} A_k \ \_k + \sum_{i=0}^{n} D_i \ \_i \qquad T(z)$. Hence, we can conclude that strong duality holds and that the primal and dual optimal values are achieved.

In order to compute the fixed point, as in §4.2, we include z as a variable for $\mathrm{SDP}_D(z)$, reintroduce the constant term that was omitted, and add a linear constraint to equate the objective function with $v_0 z$. The SDP that gives us the fixed point to $g(\ ) = v_0$ is as follows.

$$\text{minimize: } \sum_{k \in K} \_k \quad \sum_{i=0} \_i + c(z) \qquad (\mathrm{SDP}_{FP})$$

$$\text{subject to:} \quad \sum_{k \in K} A_{k\ k} + \sum_{i=0} D_{i\ i} \qquad T(z)$$
$$\sum_{k \in K} {}_{k} \sum_{i=0} {}_{i} + c(z) = v_0 z$$

Similar to the Lemma 9, if $(\hat{g}_{k \in K}; \hat{}_i g_{i=0}^{n}; \hat{z})$ is an optimal solution to SDP_{FP}, then $\hat{z}$ is a fixed point for $g(\ )=v_0$. The proof of this fact is identical to the proof of Lemma 9.

Using Lemma 19, we obtain an optimal solution to SDP(z) with $z = \hat{z}$ that does not select any vertex of N_z. So, for the rounding, it suffices to focus on the subgraph G_z with non-negative edge weights. Using the randomized 0:874 approximation guarantee from Lewin et al. (2002) that is relative to the SDP relaxation in (13)-(18), we obtain Theorem 2.

Appendix C: Missing Proofs from Section 4

Proof of Lemma 8 We first observe that $g(z)$ is continuous. For any z, as $g(z)$ is the optimal value of LP(z), it equals the maximum objective value over finitely many extreme points of the feasible region of this LP. As the objective coefficients $w_{ij}(z)$ are linear in z, it follows that $g(z)$ is a piecewise linear function, which is continuous. Moreover, since $g(z) \quad f(z)$, we have $g(z) \quad 0$ for all $z \in \mathbb{R}$. Coupled with Observation 1 and the fact that $v_0 z$ is strictly increasing in z, we are guaranteed the existence of a fixed point $\hat{z} \quad 0$ for $g(\ )=v_0$.

Proof of Lemma 9 Let z^0 be the point such that $g(z^0) = v_0 z^0$. We want to show that $\hat{z} = z^0$. Let $(\ '; \ '; \ '; \ ^0)$ be an optimal solution to the linear program described by $D(z^0)$. By strong duality for $LP(z^0)$, we have $\sum_{(i;j) \in E} {}^0_{i;j} + \sum_{i \in [n]} {}^0_i + c\ ^0 = g(z^0) = v_0 z^0$. So, $(\ '; \ '; \ '; \ ^0; z^0)$ is a feasible solution to the linear program (D). Thus, we have

$$v_0 z^0 = g(z^0) \sum_{(i;j) \in E} {}^0_{i;j} + \sum_{i \in [n]} {}^0_i + c\ ^0 \sum_{(i;j) \in E} \hat{}_{i;j} + \sum_{i \in [n]} \hat{}_i + c\ \hat{} = v_0 \hat{z};$$

which implies that $z^0 \quad \hat{z}$ (since $v_0 \quad 0$). Above, the inequality uses that $(\hat{}; \hat{}; \hat{}; \hat{})$ is an optimal solution to (D). We now show that $z^0 = \hat{z}$. Suppose, for a contradiction, $z^0 > \hat{z}$. Note that $(\hat{}; \hat{}; \hat{}; \hat{})$ is feasible to $D(\hat{z})$. By strong duality for $LP(\hat{z})$, $g(\hat{z})$ equals the optimal value of $D(\hat{z})$. So,

$$g(\hat{z}) \sum_{(i;j) \in E} \hat{}_{i;j} + \sum_{i \in [n]} \hat{}_i + c\ \hat{} = v_0 \hat{z} < v_0 z^0 = g(z^0):$$

As $g(z)$ is non-increasing in z (Observation 1), we obtain $\hat{z} > z^0$, a contradiction.

Proof of Theorem 4 Recall that $(x; y)$ is an optimal basic feasible solution to the LP (9). By the definition of a basic feasible solution, there are $n+m$ linearly independent and active constraints for $(x; y)$ where n and m are the numbers of x and y variables. Let A denote the set of all active

constraints. Note that $(x;y)$ is the unique solution for the system of linear equations in A, so A has rank $n+m$. We consider two cases.

The constraint $\sum_{i \in [n]} s_i x_i \leq c$ *is not active*. Starting from the linear system A of rank $n+m$, we iteratively reduce the rank to obtain an "equivalent" linear system of rank n. This done by carefully eliminating all the y variables. For every variable y_{ij}, we claim that either both $y_{ij} \leq x_i$ and $y_{ij} \leq 1 - x_j$ are active or exactly one of them is active. If both these constraints were inactive then we could increase y_{ij} to obtain a higher objective value, contradicting the optimality of $(x;z)$. We now modify the linear system as follows:

If both the constraints for y_{ij} are active, we replace them by the single constraint $x_i + x_j = 1$ and drop the variable y_{ij}.

If exactly one of the constraints for y_{ij} is active, we drop this constraint and y_{ij}.

In either case, the number of variables (and hence the rank of the linear system) is reduced by one. It is easy to see that the modified linear system also has a unique solution as A has a unique solution. By applying this modification for every y_{ij}, we obtain a linear system A' with exactly n variables that has a unique solution. Let us denote by E', the set of (unordered) pairs $(i;j)$ for which constraint $x_i + x_j = 1$ appears in the linear system A'. Letting $\{p_i\}_{i=1}^n$ denote the variables in A', its constraints are:

$$\begin{array}{ll} p_i + p_j = 1; \ \forall (i;j) \in E' & \\ p_i = 0; & \text{if } x_i = 0 \\ p_i = 1; & \text{if } x_i = 1 \end{array} \tag{29}$$

Note that x is the unique solution to this system. Let $I = \{i \in [n] : x_i \in \{0; 1/2; 1\}\}$ and $I' = [n] \setminus I$. Note that there are no edges in E' between I and I' : this is because the corresponding x-values cannot add to one. Let $E'' \subseteq E'$ be the set of edges incident to vertices in I : note that both end-points of these edges lie in I . The constraints in A' involving variables $\{p_i : i \in I\}$ are just:

$$p_i + p_j = 1; \quad \forall (i;j) \in E''. \tag{30}$$

As x is the unique solution to A', it follows that $\{x_i : i \in I\}$ is the unique solution to linear system (30). Now, let $E''' \subseteq E''$ be a maximal subset so that the constraints

$$p_i + p_j = 1; \quad \forall (i;j) \in E''';$$

are linearly independent. Again, $\{x_i : i \in I\}$ is the unique solution to this linear system. So $|E'''| = |I|$. We now claim that $I' = \emptyset$, which would prove the theorem. Suppose not: then the subgraph induced by the set E''' on vertices in I' must contain a cycle U because $|E''| = |I|$. Since $x_i \in \{0; \frac{1}{2}; 1\}$ for each $i \in I'$, the cycle U must have an even number of vertices, which contradicts the fact that constraints in (30) are linearly independent.

The constraint $\sum_{i\in[n]} s_i x_i \le c$ *is active.* The proof in this case essentially follows from the proof of Theorem 1 in Ageev et al. (2001). As the result in Ageev et al. (2001) was only for the special case that all sizes $s_i = 1$, we now provide a complete proof for general sizes. Using the same approach as in the previous case to drop variables $\{y_{ij}\}$, we obtain a linear system similar to (29) with n variables that has x as its unique solution:

$$\begin{array}{ll} \sum_{i=1}^{n} s_i p_i = c & \\ p_i + p_j = 1; & \forall (i,j) \in E' \\ p_i = 0; & \text{if } x_i = 0 \\ p_i = 1; & \text{if } x_i = 1 \end{array} \tag{31}$$

As before, let $I = \{i \in [n] : x_i \in \{0, 1/2, 1\}\}$, $J = [n] \setminus I$ and $E'' \subseteq E'$ be the edges incident to J. Recall that both end-points of edges in E'' lie in J. Let $c' = \sum_{i\in J} s_i x_i$. As x is the unique solution to (31), it must be that $\{x_i : i \in J\}$ is the unique solution to:

$$\begin{array}{ll} p_i + p_j = 1; & \forall (i,j) \in E'' \\ \sum_{i\in J} s_i p_i = c' & \end{array} \tag{32}$$

Now, let $E''' \subseteq E''$ be a maximal subset so that the constraints

$$\begin{array}{ll} p_i + p_j = 1; & \forall (i,j) \in E''' \\ \sum_{i\in J} s_i p_i = c' & \end{array}$$

are linearly independent. There is one caveat: it could happen that the knapsack constraint $\sum_{i\in J} s_i p_i = c'$ is not part of any maximal linearly-independent system; then, the rest of the proof is identical to the previous case (where the knapsack constraint is inactive). Below, we assume that there is a linear system as above that is linearly independent and has a unique solution. So $|J| = |E'''| + 1$. We now claim that there is no cycle in the edges E''': this follows as before because any cycle on J must be even, which contradicts the linear independence. So the subgraph (J, E''') is a tree. Choose any vertex $r \in J$ as a root. Using the equalities corresponding to E''' and the fact that $x_i \in \{0, \frac{1}{2}, 1\}$ for all $i \in J$, it follows that all vertices at odd (resp. even) distance from r have the same value. Moreover, these two values add to one. So there is some $0 \le \alpha \le \frac{1}{2}$ such that $x_i \in \{\alpha, 1-\alpha\}$ for every $i \in J$. This completes the proof.

Proof of Lemma 11 Recall that

$$F(\mathbf{x}) = \sum_{(i,j)\in E} w_{ij} x_i (1 - x_j) \quad \text{and} \quad L(\mathbf{x}) = \sum_{(i,j)\in E} w_{ij} \min\{x_i, (1 - x_j)\}.$$

From Theorem 4, since $\mathbf{x}$ is an optimal basic solution to LP (9), there is a $0 \le \alpha \le \frac{1}{2}$ such that $x_i \in \{0, \alpha, \frac{1}{2}, 1-\alpha, 1\}$ for all $i \in [n]$. Recall that

$$V_1 = \{i \mid x_i = 0\}; V_2 = \{i \mid x_i = 1-\alpha\}; V_3 = \{i \mid x_i = \frac{1}{2}\} \text{ and } V_4 = \{x_i \mid x_i = 0 \text{ or } 1\}.$$

Consider any $(i,j) \in E$ with $\min\{x_i, 1 - x_j\} > 0$, where $i \in V_p$ and $j \in V_q$. Then, we have one of the following cases:

1. $\max\{x_i; 1 - x_j\} =$ when $(p;q) \in \{(1;2)\}$
2. $\max\{x_i; 1 - x_j\} = 1$ when $(p;q) \in \{(1;1); (2;1); (2;2); (2;3); (3;1)\}$
3. $\max\{x_i; 1 - x_j\} = \frac{1}{2}$ when $(p;q) \in \{(3;2); (3;3); (1;3)\}$
4. $\max\{x_i; 1 - x_j\} = 1$ when $(p;q) \in \{(1;4); (2;4); (3;4); (4;1); (4;2); (4;3); (4;4)\}$

Any $(p;q)$ pair not listed above has $\min\{x_i; 1 - x_j\} = 0$ for all $i \in V_p$, $j \in V_q$.

Let $w_{ij} \min\{x_i; 1 - x_j\}$ be the weight contributed by edge $(i;j)$ to $L(\mathbf{x})$. We denote by $I_{pq} = \sum_{i \in V_p} \sum_{j \in V_q} w_{ij} \min\{x_i; 1 - x_j\}$ the total contributed weight of edges from V_p to V_q where $p;q \in \{1;2;3;4\}$. We group these weights according to the four cases above: $Y_1 = I_{12}$, $Y_2 = I_{11} + I_{21} + I_{22} + I_{23} + I_{31}$, $Y_3 = I_{13} + I_{32} + I_{33}$ and $Y_4 = \sum_{i=1}^{3} (I_{i4} + I_{4i}) + I_{44}$. Note that $L(\mathbf{x}) = Y_1 + Y_2 + Y_3 + Y_4$.

Using the fact that $x_i(1 - x_j) = \min\{x_i; 1 - x_j\} \max\{x_i; 1 - x_j\}$ and the four cases above,

$$F(\mathbf{x}) = \sum_{p=1}^{4} \sum_{q=1}^{4} \sum_{i \in V_p; j \in V_q} w_{ij} \min\{x_i; 1 - x_j\} \max\{x_i; 1 - x_j\} = Y_1 + (1 - \alpha) Y_2 + \frac{1}{2} Y_3 + Y_4.$$

We assert that $F(\mathbf{x}^0) \geq Y_1 + \frac{Y_3}{2}$ where $\mathbf{x}^0$ is defined in (10). Note that $x^0_i = x_i$ for all $i \in V_3 \cup V_4$. We now consider two cases. First, suppose $S(V_1) \leq S(V_2)$ where $S(V_p) = \sum_{i \in V_p} s_i$. Then, by (10) we have $x^0_i = 0$ for all $i \in V_2$, and $x^0_i \geq x_i =$ for all $i \in V_1$. It can be checked directly that $\max\{x^0_i; 1 - x^0_j\} \min\{x^0_i; 1 - x^0_j\} \geq \max\{x_i; 1 - x_j\} \min\{x_i; 1 - x_j\}$ for all $i \in V_p; j \in V_q$ where $(p;q) \in \{(1;2); (3;2); (3;3); (1;3)\}$. Moreover, for $i \in V_1$ and $j \in V_2$, we have $\max\{x^0_i; 1 - x^0_j\} = 1$ whereas $\max\{x_i; 1 - x_j\} =$. So, $F(\mathbf{x}^0) \geq Y_1 + \frac{1}{2} Y_3$. On the other hand, if $S(V_1) \geq S(V_2)$, we have $x^0_i = 1$ for all $i \in V_1$ and $x^0_i \geq x_i = 1$ for all $i \in V_2$. Using similar calculations as the previous case, we again obtain $F(\mathbf{x}^0) \geq Y_1 + \frac{1}{2} Y_3$.

Let $\alpha = \frac{1-\alpha/2}{1}$; note that $\alpha \in [0;1]$. It now follows that

$$\begin{aligned}
\max\{F(\mathbf{x}); F(\mathbf{x}^0)\} &\geq \max\{Y_1;\ Y_1 + (1 - \alpha) Y_2 + Y_4\} + \frac{Y_3}{2} \geq \max\{Y_1;\ Y_1 + (1 - \alpha)(Y_2 + Y_4)\} + \frac{Y_3}{2} \\
&\geq Y_1 + (1 - \alpha) \left(Y_1 + (1 - \alpha)(Y_2 + Y_4) \right) + \frac{Y_3}{2} \\
&= (\alpha + (1 - \alpha)) Y_1 + (1 - \alpha)(1 - \alpha)(Y_2 + Y_4) + \frac{Y_3}{2} = \frac{1}{2}(Y_1 + Y_2 + Y_4 + Y_3) = 0.5 L(\mathbf{x}).
\end{aligned}$$

Recall that the α convexity property holds for F. Since we run pipage rounding (Theorem 3) with *both* $\mathbf{x}$ and $\mathbf{x}^0$ and pick the better solution, we obtain a solution of weight at least $0.5\ L(\mathbf{x})$ which concludes the proof. Because $\sum_i x_i = \sum_i x^0_i = c$, Theorem 3 also implies that both the rounded solutions are integral.

Proof of Lemma 12 Recall that $\mathbf{x}$ is the basic optimal solution of (9). As noted before, F satisfies the α-convexity property. In order to apply Theorem 3 we will show that the F/L lower bound property holds for $\mathbf{x}$ with $\alpha = 0.5$. As constraint $\sum_i s_i x_i \leq c$ is not active, by Theorem 4

we know that $x_i \in \{0, \frac{1}{2}, 1\}$ for all $i \in V$. So for every $(i,j) \in E$ that contributes a positive value to $L(\mathbf{x})$, we have $\max\{x_i, 1 - x_j\} \geq \frac{1}{2}$. Using the half-integrality of $\mathbf{x}$, for any $(i,j) \in E$ we have

$$x_i(1 - x_j) + \min\{x_i, 1 - x_j\} \cdot \max\{x_i, 1 - x_j\} \geq \frac{1}{2} \cdot \min\{x_i, 1 - x_j\}.$$

So the contribution of any $(i,j) \in E$ in F is at least $\frac{1}{2}$ times its contribution in L. Hence $F(\mathbf{x}) \geq 0.5 \cdot L(\mathbf{x})$, which concludes the proof.

Proof of Theorem 5 In this proof, variables $\mathbf{x}$, $\mathbf{x}^0$, $\hat{\mathbf{x}}$, and $\mathbf{x}'$ correspond to those defined in Algorithm 1. By Theorem 3 and Lemmas 11 and 12, it follows that solution $\mathbf{x}'$ has at most one fractional variable and $F(\mathbf{x}') \geq F(\hat{\mathbf{x}}) \geq 0.5 \cdot L(\mathbf{x})$. We now consider two cases.

If constraint $\sum_i x_i \leq c$ is active then we know that $\sum_i \hat{x}_i = c$; note that $\sum_i x^0_i = \sum_i x_i = c$. In this case, Theorem 3 implies that $\mathbf{x}'$ is integral. So the resulting solution I_1 has value $F(\mathbf{x}') \geq 0.5 \cdot L(\mathbf{x})$.

If constraint $\sum_i x_i \leq c$ is not active then Theorem 4 implies that $x_i \in \{0, \frac{1}{2}, 1\}$ for all $i \in V$. Let $\sum_i x_i = c' < c$. Again, if $\mathbf{x}'$ is itself integral then solution I_1 has value $F(\mathbf{x}') \geq 0.5 \cdot L(\mathbf{x})$. It remains to consider the case that $\mathbf{x}'$ is not integral. Let $i \in V$ denote the only fractional variable in $\mathbf{x}'$. Note that $\sum_i x_i = \sum_i \hat{x}_i = \sum_i x_i = c'$. As $\mathbf{x}'$ has a unique fractional variable, c' is not integer. So $|I_1| = \lfloor c' \rfloor \leq c$ and $|I_1| + 1 = \lceil c' \rceil \leq c$. Hence, both the solutions I_1 and $I_1 \cup \{i\}$ satisfy the capacity constraint. In order to bound the objective let us define the vertex-sets $U_0 = \{j \in V \mid x_j = 0\}$, and $U_1 = \{j \in V \mid x_j = 1\}$. Let W_{10} be the total weight of edges from U_1 to U_0, W_0 the total weight of edges from i to U_0 and W_1 the total weight of edges from U_1 to i. The cut value of $\mathbf{x}'$ is $F(\mathbf{x}') = W_{10} + \frac{W_0}{2} + \frac{W_1}{2} \geq W_{10} + \max\{W_0, W_1\}$. The two integral solutions we consider have values $W_{10} + W_0$ and $W_{10} + W_1$: so the better of these has value at least $F(\mathbf{x}') \geq 0.5 \cdot L(\mathbf{x})$.

So in all cases we obtain a solution of value at least $0.5 \cdot L(\mathbf{x})$, which proves the approximation ratio for max-dicut under capacity constraints.

Using this rounding algorithm within the approach in §4.3 and Theorem 1, we also obtain a 0.5-approximation algorithm for the capacitated assortment problem.

Proof of Theorem 6 If $\sum_i s_i x_i = c$ then step 2 applies and $F(\hat{\mathbf{x}}) = \max\{F(\mathbf{x}), F(\mathbf{x}^0)\} \geq 0.5 \cdot L(\mathbf{x})$ by Lemma 11. If $\sum_i s_i x_i < c$ then step 3 applies and we have $F(\hat{\mathbf{x}}) = F(\mathbf{x}') \geq 0.5 \cdot L(\mathbf{x})$ by Lemma 12. In either case, we have $F(\hat{\mathbf{x}}) \geq 0.5 \cdot L(\mathbf{x})$.

Now, using Theorem 3, it follows that $F(\mathbf{x}') \geq F(\hat{\mathbf{x}}) \geq 0.5 \cdot L(\mathbf{x})$. We now show that the better of solutions I_1 and I_2 in step 4 has weight at least $\frac{1}{2}F(\mathbf{x}')$ which would prove the theorem. To see this, let $x_i = f$ be the fractional variable. As in the proof of Theorem 5, define sets of vertices $U_0 = \{j \in V \mid x_j = 0\}$, and $U_1 = \{j \in V \mid x_j = 1\}$. Let W_{10} be the total weight of edges from U_1 to U_0, W_0 the total weight of edges from i to U_0 and W_1 the total weight of edges from U_1 to i. Clearly $F(\mathbf{x}') = W_{10} + f \cdot W_0 + (1 - f)W_1$. The cut value from solution I_1 (resp. I_2) is at least $W_{10} + W_1$ (resp W_0). So the better of these two is at least $\max(W_{10} + W_1, W_0) \geq \frac{1}{2}(W_{10} + W_1 + W_0) \geq \frac{1}{2}(W_{10} + f W_0 + (1 - f)W_1) = 0.5F(\mathbf{x}')$ because $f \in [0, 1]$.

Appendix D: Missing Proofs from Section 5

Proof of Theorem 7 We first need a few useful lemmas.

Lemma 21. *The expected revenue of the optimal assortment $z \in [R_{min}; R_{max}]$, where $R_{min} = \frac{\min_i(r_i)\ \min_i(v_i)}{2\ \max_i(v_i)}$ and $R_{max} = \max_i(r_i)$.*

Proof of Lemma 21. Recall that we have defined z in Equation (2) as

$$z = (\mathbf{x}) = \max_{\mathbf{x} \in F} (\mathbf{x}) = \frac{\sum_{(i;j) \in M} V_{ij}(\mathbf{x})\ ^{ij} R_{ij}(\mathbf{x})}{v_0 + \sum_{(k;l) \in M} V_{kl}(\mathbf{x})\ ^{kl}}.$$

We also have the following upper bound on $R_{ij}(\mathbf{x})$:

$$R_{ij}(\mathbf{x}) = \frac{r_i v_i^{1=\ ^{ij}} x_i + r_j v_j^{1=\ ^{ij}} x_j}{v_i^{1=\ ^{ij}} x_i + v_j^{1=\ ^{ij}} x_j} \quad \frac{R_{max} v_i^{1=\ ^{ij}} x_i + R_{max} v_j^{1=\ ^{ij}} x_j}{v_i^{1=\ ^{ij}} x_i + v_j^{1=\ ^{ij}} x_j} = R_{max}.$$

Thus, we conclude that $z \quad R_{max}$. To obtain a lower bound on z, consider the assortment which consists of displaying only product 1 (Without loss of generality, assume that product 1 has $r_1 > 0$ and $v_1 > 0$), i.e. we set $x_1 = 1$ and $x_i = 0$ for all $i \in [n] \setminus \{1\}$. Then,

$$z = \frac{\sum_{(i;j) \in M} V_{ij}(\mathbf{x})\ ^{ij} R_{ij}(\mathbf{x})}{v_0 + \sum_{(k;l) \in M} V_{kl}(\mathbf{x})\ ^{kl}} = \frac{\sum_{j \in N} v_1 r_1}{v_0 + \sum_{j \in N} v_1} \quad \frac{n v_1 r_1}{(n+1)\max_i(v_i)} \quad \frac{r_1 v_1}{2\ \max_i(v_i)} \quad \frac{\min_i(r_i)\ \min_i(v_i)}{2\ \max_i(v_i)}$$

where the first inequality follows by replacing v_0 and v_1 by $\max_i(v_i)$ and the second inequality follows because $2n \quad n+1$ for all $n \quad 1$.

Lemma 22. *If $ALG_z \quad v_0 z$, then $z \quad z$.*

Proof of Lemma 22. Let $\mathbf{x}$ be the optimal assortment that achieves an expected revenue of z, i.e. $(\mathbf{x}) = z$. We have that

$$ALG_z = \sum_{(i;j) \in M} V_{ij}(\hat{\mathbf{x}})\ ^{ij} [R_{ij}(\hat{\mathbf{x}}) \quad z] \quad v_0 z.$$

Rearranging the terms gives us

$$\frac{\sum_{(i;j) \in M} V_{ij}(\hat{\mathbf{x}})\ ^{ij} R_{ij}(\hat{\mathbf{x}})}{v_0 + \sum_{(i;j) \in M} V_{ij}(\hat{\mathbf{x}})\ ^{ij}} \quad z.$$

The term on the LHS is exactly $(\hat{\mathbf{x}}) \quad (\mathbf{x})$, and thus we conclude that $z \quad z$.

Lemma 23. *If $ALG_z < v_0 z$, then $z < \frac{z}{c}$.*

Proof of Lemma 23. We prove the above claim using its contrapositive. We want to show that if $z \quad \frac{z}{c}$, then $ALG_z \quad v_0 z$. Let $\mathbf{x}$ be the optimal assortment that achieves an expected revenue of z, i.e. $(\mathbf{x}) = z$, and $\tilde{\mathbf{x}}$ be the assortment that maximizes $f(z)$. From $z \quad \frac{z}{c}$, we get

$$\frac{\sum_{(i;j) \in M} V_{ij}(\mathbf{x})\ ^{ij} R_{ij}(\mathbf{x})}{v_0 + \sum_{(i;j) \in M} V_{ij}(\mathbf{x})\ ^{ij}} \quad \frac{z}{c}.$$

Rearranging terms to get,

$$c \sum_{(i;j)2M} V_{ij}(\mathbf{x}\ )\ ^{ij} R_{ij}(\mathbf{x}\ ) \quad v_0 z + z \sum_{(i;j)2M} V_{ij}(\mathbf{x}\ )\ ^{ij} \quad v_0 z + z\ c \sum_{(i;j)2M} V_{ij}(\mathbf{x}\ )\ ^{ij}$$

where the second inequality follows because c 1. This gives us the relation

$$c \sum_{(i;j)2M} V_{ij}(\mathbf{x}\ )\ ^{ij}[R_{ij}(\mathbf{x}\ ) \quad z] \quad v_0 z:$$

From the definition of $\tilde{\mathbf{x}}$, the equivalence of $f(z)$ and the max-dicut instance (Lemma 6) and the fact that our solution $\hat{\mathbf{x}}$ is an c approximation to the optimal, we get

$$ALG_z = \sum_{(i;j)2M} V_{ij}(\hat{\mathbf{x}})\ ^{ij}[R_{ij}(\hat{\mathbf{x}}) \quad z] \quad c \sum_{(i;j)2M} V_{ij}(\tilde{\mathbf{x}})\ ^{ij}[R_{ij}(\tilde{\mathbf{x}}) \quad z] \quad c \sum_{(i;j)2M} V_{ij}(\mathbf{x}\ )\ ^{ij}[R_{ij}(\mathbf{x}\ ) \quad z] \quad v_0 z$$

which concludes the proof.

Lemma 24. *At every iteration of Algorithm 2,* $L \quad z < \frac{R}{c}$.

Proof of Lemma 24. This is clearly true in the first iteration. Inductively, assume that the claim is true for some intermediate iteration, say iteration k. We want to show that the claim continues to hold in iteration $k+1$. If $ALG_z \quad v_0 z$, then Algorithm 2 sets $L \quad z$, and iterates. By the induction hypothesis, $z \quad \frac{R}{c}$ continues to hold, and by Lemma 22, $L \quad z$. On the other hand, if $ALG_z < v_0 z$, then Algorithm 2 sets $R \quad z$, and iterates. By the induction hypothesis, $L \quad z$ continues to hold, and by Lemma 23, $z < \frac{R}{c}$. This concludes the induction.

We are now ready to complete the proof of Theorem 7. First, assume that L has been updated at least once in the course of the algorithm. Then, there must be an assortment $\hat{\mathbf{x}}$ found in Step 4 of some iteration where $\sum_{(i;j)2M} V_{ij}(\hat{\mathbf{x}})\ ^{ij}[R_{ij}(\hat{\mathbf{x}}) \quad L] \quad v_0 L$. On rearranging terms, we get

$$\frac{\sum_{(i;j)2M} V_{ij}(\hat{\mathbf{x}})\ ^{ij} R_{ij}(\hat{\mathbf{x}})}{v_0 + \sum_{(i;j)2M} V_{ij}(\hat{\mathbf{x}})\ ^{ij}} \quad L$$

where the left-hand-side of the inequality is the expected revenue $(\hat{\mathbf{x}})$ of our solution. If L was never updated, then $L = R_{min}$ and we can return the assortment used to calculate R_{min}. The above equation continues to hold for this case. Using Lemma 24 we know that $z < \frac{R}{c}$, and from the termination condition we know $R \quad L$. Putting these together, we get $z < \frac{L+}{c}$. Thus,

$$(\tilde{x}) \quad L \quad cz \quad = \quad cz \quad R_{min} \quad (\ c \quad)z$$

which is our desired approximation guarantee.

Let the number of iterations of Algorithm 2 be t. At each step, we halve the range $R \quad L$ for the search, and the search stops when this range is smaller than $= R_{min}$. So Algorithm 2 terminates

after $t = \log_2 \frac{R_{max} - R_{min}}{R_{min}}$ iterations, which is polynomial in the input. Moreover, the time taken in each iteration is dominated by the approximation algorithm for max-dicut which is denoted $p(n)$. So Algorithm 2 runs in time $O(\log \frac{R_{max} - R_{min}}{R_{min}} \cdot p(n))$. Theorem 7 now follows by using the values R_{min} and R_{max} from Lemma 21.

Proof of Theorem 8 Note that the overall enumeration needed is $n^{1+4=\ 0} \cdot O(\log n)$ for subset S and value U. Also, for each choice of U and S, the algorithm runs in polynomial time. So the overall algorithm runs in polynomial time.

We now prove the approximation ratio. Note that $OPT \quad U \quad 2 \cdot OPT$ for some choice of U. Let T denote the set of heavy vertices with respect to $\ ^0$ that are in some optimal solution. By Lemma 13, we know that $|T| \quad d4=\ ^0 e$. So we will have subset $S = T$ in some enumeration. For this choice, it is clear that $LP\ (S) \quad OPT$ and we obtain a cut of value at least $0{:}5 \cdot OPT \quad \ ^0 U \quad (0{:}5 \quad) \cdot OPT$. So we obtain an $(0{:}5 \quad)$-approximation algorithm for max-dicut with a knapsack constraint. The second statement follows by combining this result with Theorem 7.

Appendix E: Missing Proofs from Section 6

Proof of Lemma 15. Fix $i \in Q_z$. Define random variable Z_k for $k \in \overline{OPT}$ as

$$Z_k = \begin{cases} \ _{ik}=2; & \text{with probability } p \\ 0; & \text{otherwise} \end{cases}.$$

Then we have $\ _i = 2 \sum_{k \in \overline{OPT}} Z_k$ and $\mathsf{E}_S[\ _i] = p \cdot O_i$. Note that each $Z_k \in [0;1]$. For each k, let $X_k = Z_k \quad \mathsf{E}[Z_k]$; note that $|X_k| \quad 1$ and $\mathsf{E}[X_k] = 0$. Because $\{X_k : k \in \overline{OPT}\}$ are independent, Bernstein's inequality gives:

$$\mathbf{P}\left[\left|\sum_k X_k\right| > t\right] \quad 2\exp\left(\frac{-t^2}{2\sum_k \mathsf{E}[X_k^2] + 2t=3}\right); \quad \text{for any } t > 0.$$

Let $= \sum_k \mathsf{E}[Z_k] = pO_i=2$ and $t = \ Np=2$. Note that $\sum_k \mathsf{E}[X_k^2] \quad \sum_k \mathsf{E}[|X_k|] \quad 2\sum_k \mathsf{E}[Z_k] = 2 \ .$

$$\mathbf{P}[|\ _i \quad \mathsf{E}[\ _i]| > \ Np] = \mathbf{P}\left[\left|\sum_k Z_k \quad \right| > \frac{Np}{2}\right] = \mathbf{P}\left[\left|\sum_k X_k\right| > t\right] \quad 2\exp\left(\frac{-t^2}{4 \ + 2t=3}\right) \tag{33}$$

Using the values of and t,

$$\frac{t^2}{4 \ + 2t=3} \quad \min\left(\frac{t^2}{8}; \frac{t^2}{4t=3}\right) = \min\left(\frac{\ ^2N^2p}{16O_i}; \frac{3 \ Np}{8}\right) \quad \min\left(\frac{\ ^2Np}{32}; \frac{3 \ Np}{8}\right) = \frac{\ ^2Np}{32} = 3\log N.$$

The last inequality used the fact that $O_i \quad 2N$ and the last equality used the definition of p. Combined with (33) we obtain $\mathbf{P}[|\ _i \quad \mathsf{E}[\ _i]| > \ Np] \quad \frac{2}{N^3}$.

Proof of Lemma 16. We will show that $\mathbf{P}_S[LP(\hat{\mathbf{x}}) \quad OPT \quad N \sum_{i \in Q_z} v_i r_i] \quad 1 \quad \frac{3}{N^2}$. Clearly, this implies the lemma as the objective is always non-negative. Let B denote the event that $|S|$

$2Np$, which represents the non-trivial case in the algorithm. By Chernoff bound, $\mathbf{P}[B] \quad 1 \quad e^{\ Np=3} \quad 1 \quad \frac{1}{N^8}$.

We condition on the events B and that $_i\ 2\ [p \quad O_i \quad\quad Np]$ for all $i\ 2\ Q_z$. By Lemma 15 and a union bound, this happens with probability at least $1 \quad \frac{2}{N^2} \quad \frac{1}{N^8} \quad 1 \quad \frac{3}{N^2}$. We now have $\frac{i}{p} \quad N \quad O_i = \sum_{j\ 2\ \overline{OPT}} \ _{ij}$ for all i. This implies that the integral solution OPT satisfies all constraints of LP( ). Moreover, the objective value of solution OPT in LP( ) is:

$$\frac{1}{p} \sum_{i2OPT} \ _i v_i r_i + \sum_{i2OPT} C_i(z) \quad \sum_{i2OPT} (O_i \quad N\ )v_i r_i + \sum_{i2OPT} C_i(z) = OPT \quad N \sum_{i2OPT} v_i r_i;$$

which is at least $OPT \quad N \sum_{i2Q_z} v_i r_i$. Thus, with probability at least $1 \quad \frac{3}{N^2}$, we have a feasible solution to LP( ) of objective at least $OPT \quad N \sum_{i2Q_z} v_i r_i$.

Proof of Lemma 17. Recall that $\hat{\mathbf{x}}$ is an optimal solution to LP( ). The objective of $\mathbf{x}$ in (11) is:

$$\sum_{i;j2Q_z} v_i r_i \ _{ij} \ \mathbf{1}[x_i = 1; x_j = 0] + \sum_{i2Q_z} C_i(z) \ \mathbf{1}[x_i = 1]:$$

Conditioned on the choice of S,

$$\begin{aligned}
\mathsf{E}[ALG|S] &= \sum_{i;j2Q_z} v_i r_i \ _{ij} \ \mathbf{P}[x_i = 1; x_j = 0] + \sum_{i2Q_z} C_i(z) \ \mathbf{P}[x_i = 1] \\
&= \sum_{i2Q_z} v_i r_i \hat{x}_i \sum_{j2Q_z} \ _{ij}(1 \quad \hat{x}_j) + \sum_{i2Q_z} C_i(z)\hat{x}_i \\
&\quad \sum_{i2Q_z} v_i r_i \hat{x}_i \quad \frac{i}{p} \quad N \quad + \sum_{i2Q_z} C_i(z)\hat{x}_i \quad LP(\hat{\mathbf{x}}) \quad N \sum_{i2Q_z} v_i r_i:
\end{aligned}$$

where the first inequality follows from the feasibility of $\hat{\mathbf{x}}$ for LP( ). Taking expectation over S and using Lemma 16,

$$\mathsf{E}[ALG] \quad 1 \quad \frac{3}{N^2} \ OPT \quad 2N \sum_{i2Q_z} v_i r_i \quad 1 \quad \frac{3}{N^2} \ OPT \quad 6 \ OPT;$$

where the last inequality follows from Lemma 14.

E.1. PTAS for Capacitated Assortment Optimization

Here, we consider the capacitated problem when $c \quad n$ for some constant $\quad > 0$ and prove Theorem 12. The proof is similar to that of the unconstrained problem (Theorem 10) and uses the binary-search framework (Theorem 7). We will show that for any $z \quad 0$, there is a randomized PTAS for the function value $f(z)$ in (3) when the feasible solutions are $F = \{\mathbf{x}\ 2\ \{0;1\}^n : \sum_{i=1}^{n} x_i \quad c\}$. Recall from §3 and Lemmas 5 and 6, that the function value $f(z)$ equals the optimal max-dicut value on $\hat{G}_z$ subject to the capacity constraint (and that the dummy vertex $n+1$ is not selected).

As before, $\hat{G}_z$ is the subgraph of G_z consisting of non-negative edges. In our reduction described in §3, recall that for every pair $(i;j)\ 2\ [n] \quad [n]$, we add four edges: $(i;n+1)$, $(j;n+1)$, $(i;j)$ and $(j;i)$

with weights $^{+}_{ij}$; $_{ji}$; $^{+}_{ij}$, and $_{ji}$ respectively. From the definitions (4) and (5), we have $^{+}_{ij} + ^{+}_{ij} = v_i\bar{r}_i$ and $_{ji} + _{ji} = v_j\bar{r}_j$. Also recall the edge weights $w_{ij}(z)$ as defined in (6). In particular, for $i; j \in [n]$ we have $w_{ij}(z) = ^{+}_{ij} + _{ij}$. Let $Q_z = \{i \in [n] : \bar{r}_i \geq 0\}$; recall that $P_z = Q_z \cup \{n+1\}$ and $N_z = [n+1] \setminus P_z$. Let $N = |Q_z|$. For any $i \in Q_z$ and $j \in [n]$, we have $w_{ij}(z) \geq 2v_i\bar{r}_i$ based on the calculations in Lemma 2. For easier notation, for all $i; j \in Q_z$ we define $_{ij} = \frac{w_{ij}(z)}{v_i\bar{r}_i} \in [0; 2]$. Also, for any $i \in Q_z$, define $C_i(z) = \sum_{j \in Q_z} w_{ij}(z)$. Finally, note that the optimal max-dicut on $\mathcal{G}_z$ only contains vertices from Q_z. So $f(z)$ equals:

$$\max_{A \subseteq Q_z; |A| \leq c} \left(\sum_{i \in A} v_i\bar{r}_i \sum_{j \in Q_z \setminus A} {}_{ij} + \sum_{i \in A} C_i(z) \right). \tag{34}$$

Below, we use OPT to refer to the optimal solution to (34) as well as its value. It will be clear from context, which of the two is being referenced. Let $\overline{OPT} = Q_z \setminus OPT$, i.e. the set of vertices that are not in the optimal max-dicut solution. For every vertex $i \in Q_z$, define $O_i = \sum_{j \in \overline{OPT}} {}_{ij}$; note that $O_i \leq 2N$. Consequently, the optimal value can be written as

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

The overall algorithm is similar to that in §6 for unconstrained assortment optimization. The PTAS starts by sampling a random subset $S \subseteq Q_z$, and then for each partition $(U; \overline{U})$ of S, it does the following and picks the best solution found. (1) Define $_i = \sum_{j \in U} {}_{ij}$ for each $i \in Q_z$. (2) Solve $\mathrm{LP}_C(\ )$ described below. (3) Randomly round the solution of $\mathrm{LP}_C(\ )$ to obtain an integral solution $R \subseteq Q_z$. (4) Obtain a *feasible* solution $R' \subseteq R$ (i.e. $|R'| \leq c$) by carefully dropping some nodes in R. The main difference from the unconstrained case is in the last step above, which ensures that the final solution satisfies the capacity constraint. The additional step in the analysis is to show that this "pruning" step does not lose much in the objective (see Lemma 28).

Theorem 14. *For any $\in (0; 1)$, Algorithm 4 runs in $n^{O(1/^2)}$ time and finds a feasible solution with expected objective at least $1 - \frac{34}{}$ times the optimal capacitated max-dicut value on $\mathcal{G}_z$, where the capacity limit $c = n$.*

We first discuss the running time of Algorithm 4. If the algorithm continues beyond step 2, the sample size $|S| \leq 2Np \leq \frac{192 \log N}{^2}$. So the number of iterations of the for-loop, which is the number of possible splits of S is at most $2^{|S|} = n^{O(1/^2)}$. Moreover, each iteration of the for-loop involves solving an LP with N variables and constraints, randomized rounding and partitioning the rounded solution, all of which takes polynomial time. So the overall runtime is $n^{O(1/^2)}$.

We now analyze the performance guarantee. Throughout, we will assume that $N \geq \frac{4}{}$: otherwise (34) can be easily solved by enumeration.

Algorithm 4 Randomized PTAS for capacitated max-dicut on $\mathcal{G}_z$

1: Sample S: for every $i \in Q_z$, add i to S independently with probability $p = \frac{96 \log N}{\epsilon^2 N}$.

2: If $|S| > 2Np$ then set $\hat{\mathbf{x}} = \mathbf{x}^* = \mathbf{0}$ and return solution $\mathbf{x}^*$.

3: **for** all splits $(U, \overline{U})$: $U \subseteq S$, $\overline{U} = S \setminus U$ **do**

4: Let $\gamma_i := \sum_{j \in \overline{U}} w_{ij}$ for all $i \in Q_z$

5: Solve the following LP and let $\hat{\mathbf{x}}$ be its optimal solution.

$$\begin{aligned} \text{maximize: } & \frac{1}{p} \sum_{i \in Q_z} \gamma_i v_i \tau_i \, x_i + \sum_{i \in Q_z} C_i(z) \, x_i \\ \text{subject to: } & \sum_{j \in Q_z} w_{ij} \, (1 - x_j) \le \frac{\gamma_i}{p} - N, \ \forall i \in Q_z \\ & \sum_{i \in Q_z} x_i \le c \\ & x_i \in [0, 1], \qquad \forall i \in Q_z \end{aligned} \tag{$\mathrm{LP}_C(\gamma)$}$$

6: Set solution $\tilde{\mathbf{x}}$ as: $\tilde{x}_i = \begin{cases} 1, & \text{with probability } \hat{x}_i \\ 0, & \text{otherwise} \end{cases}$; for all $i \in Q_z$. Let $R = \{i \in Q_z : \tilde{x}_i = 1\}$.

7: If $|R| > (1+\epsilon)c$, where $\epsilon = 3\epsilon$, then set $\mathbf{x}^* = \mathbf{0}$ and continue.

8: Else, partition R into $\ell = \lceil \frac{|R|}{c} \rceil$ parts, $R_1, R_2, \ldots, R_\ell$, with each part having at most c nodes.

9: For each $k \in [\ell]$ let $\mathrm{cut}(R_k) = \sum_{i \in R_k} v_i \tau_i \sum_{j \in Q_z \setminus R} w_{ij} + \sum_{i \in R_k} C_i(z)$. Renumber the parts so that $\mathrm{cut}(R_1) \ge \mathrm{cut}(R_2) \ge \cdots \ge \mathrm{cut}(R_\ell)$.

10: Set $R^* = \begin{cases} R, & \text{if } |R| \le c \\ \bigcup_{k=1}^{2} R_k, & \text{otherwise} \end{cases}$ and $\mathbf{x}^*$ to be the indicator vector of R^*.

11: Return the best $\mathbf{x}^*$ found over all runs.

Lemma 25. $OPT \ge \frac{\epsilon(N-1)}{8} \sum_{i \in Q_z} v_i \tau_i \ge \frac{\epsilon N}{12} \sum_{i \in Q_z} v_i \tau_i$.

Proof. Consider a random solution $S \subseteq Q_z$ to (34) generated as follows. First, each vertex $i \in Q_z$ is chosen into a set R with probability $\frac{\epsilon}{4}$ independently. Then, we set $S = R$ if $|R| \le N$, and $S = \emptyset$ otherwise. Using the definition of $C_i(z)$, the objective value of S is:

$$obj(S) = \sum_{i \in S} \Big(\sum_{j \in Q_z \setminus S} w_{ij}(z) + \sum_{j \in Q_z} w_{ij}(z) \Big);$$

For any nodes $i \in Q_z$ and j (possibly empty) we claim:

$$\mathbf{P}[jRj \quad N \mid i \in R; j \notin R] \quad \frac{1}{2}. \tag{35}$$

This follows by Markov's inequality and $\mathsf{E}[jRj \mid i \in R; j \notin R] \quad 1 + \frac{1}{4}\ (N \quad 1) < 1 + \frac{N}{4} \quad \frac{N}{2}$, where we used $N \quad 4$. Using (35),

$$\mathbf{P}[i \in S] = \mathbf{P}[i \in R; jRj \quad N] = \mathbf{P}[i \in R]\ \mathbf{P}[jRj \quad N \mid i \in R] \quad \frac{1}{4}\ \frac{1}{2} = \frac{1}{8}; \quad \forall i \in Q_z.$$

Furthermore, for any $i, j \in Q_z$, we have $\mathbf{P}[i \in S; j \notin S]$ equals:

$$\mathbf{P}[i \in R; j \notin R; jRj \quad N] = \mathbf{P}[i \in R]\ \mathbf{P}[j \notin R]\ \mathbf{P}[jRj \quad N \mid i \in R; j \notin R] \quad \frac{1}{4}\ (1 \quad \frac{1}{4})\ \frac{1}{2} \quad \frac{3}{32}.$$

We are now ready to now bound $\mathsf{E}[obj(S)]$.

$$\begin{aligned}
\mathsf{E}[obj(S)] \quad & \frac{3}{32} \sum_{i,j \in Q_z} w_{ij}(z) + \frac{1}{8} \sum_{i \in Q_z, k \notin Q_z} w_{ik}(z) \quad \frac{1}{16} \sum_{i \in Q_z} \left(\sum_{j \in Q_z} w_{ij}(z) + w_{i,n+1}(z) \right) \\
&= \frac{1}{16} \sum_{i \in Q_z} \left(\sum_{j \in Q_z} w_{ij}(z) + \sum_{k \in [n] \setminus \{i\}} (x^+_{ik} + x_{ik}) \right) \quad \frac{1}{16} \sum_{i,j \in Q_z} (x^+_{ij} + x_{ij}) + (x^+_{ij} + x_{ij}) \qquad (36) \\
&= \frac{(N \quad 1)}{16} \sum_{i \in Q_z} 2 v_i \bar{r}_i.
\end{aligned}$$

Above, the equality in (36) uses the definition of $w_{i,n+1}(z)$ and the inequality in (36) uses the definition of $w_{i,j}(z)$ and that all the terms are non-negative. The final equality uses the observation that $x^+_{ij} + x^+_{ij} = v_i \bar{r}_i = x_{ij} + x_{ij}$ from (5). The lemma now follows as $OPT \quad \mathsf{E}[obj(S)]$.

For any sample $S \quad Q_z$, define its *canonical split* as $(T_S; \overline{T_S})$ of S where $T_S = S \setminus OPT$ and $\overline{T_S} = S \setminus \overline{OPT}$. 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 $_i = \sum_{j \in S \setminus \overline{OPT}} \ _{ij}$ and so $\mathsf{E}_S[\ _i] = \sum_{j \in \overline{OPT}} \ _{ij}\ \mathbf{P}[j \in S] = p\ O_i$.

Lemma 26. $\Pr_S[j \ _i \quad \mathsf{E}[\ _i]| > \ Np] \quad \frac{2}{N^3}$ *for all* $i \in Q_z$.

So, we have $\ _i \in [p\ O_i \quad Np]$, 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 $\mathrm{LP}_C(\ )$ for the split $(T_S; \overline{T_S})$, and $LP(\hat{\mathbf{x}})$ be its value.

Lemma 27. $\mathsf{E}_S[LP(\hat{\mathbf{x}})] \quad (1 \quad \frac{3}{N^2})OPT \quad N \sum_{i \in Q_z} v_i \bar{r}_i$.

The proofs of Lemmas 26 and 27 are identical to those for Lemmas 15 and 16 respectively, and are omitted. Let ALG denote the objective value in (34) of our algorithm's solution $\mathbf{x}$.

Lemma 28. *Solution* $\mathbf{x}$ *is always feasible and* $\mathsf{E}_S[ALG] \quad (1 \quad \frac{34}{\ })\ OPT$.

Proof. We first show the feasibility. If $|R| > (1+\ )c$ then $\mathbf{x} = \mathbf{0}$ which is feasible. If $|R| \ \ c$ then $\sum_{i=1}^{n} x_i = |R| \ \ c$. If $c < |R| \ \ (1+\ )c$ then $\sum_{i=1}^{n} x_i = |R| \ \ (\ \ 2) \ \ c$ as each part R_k has at most $\ \ c$ nodes. Moreover, $\ \ \frac{|R|}{c} + 1 \ \ \frac{1}{\ } + 2$, which implies $\sum_{i=1}^{n} x_i \ \ c$.

We now analyze the objective value ALG *conditioned on some sample* S. Note that this also fixes the optimal solution $\hat{\mathbf{x}}$ of $\mathrm{LP}_C(\ )$. If $|R| \ \ c$ then $ALG = obj(R)$. If $c < |R| \ \ (1+\ )c$, then by definition of $cut(R_k)$ in step 9,

$$ALG \ \ \sum_{k=1}^{\ } cut(R_k) \ \ \frac{2}{\ } \sum_{k=1}^{\ } cut(R_k) = \frac{2}{\ } obj(R) \ \ (1 \ \ 2\ ) \ \ obj(R).$$

The second inequality above is by $cut(R_1) \ \ \ \ cut(R_{\ })$, the equality uses the definition of $cut(R_k)$ and $obj(R)$, and the last inequality uses $\ \ \frac{|R|}{c} > \frac{1}{\ }$.

Let G denote the event $|R| \ \ (1+\ )c$ and $\mathbf{1}_G$ its indicator. So, we have

$$\mathsf{E}[ALG] \ \ \mathsf{E}[ALG \ \ \mathbf{1}_G] \ \ (1 \ \ 2\ ) \ \mathsf{E}[obj(R) \ \ \mathbf{1}_G].$$

Using (34), we now have $\mathsf{E}[obj(R) \ \ \mathbf{1}_G]$ equals:

$$\sum_{i,j \in Q_z} v_i r_i \ \ _{ij} \ \ \mathbf{P}[i \in R, j \notin R, G] + \sum_{i \in Q_z} C_i(z) \ \ \mathbf{P}[i \in R, G]$$

We will show that:

$$\mathbf{P}[i \in R, j \notin R, G] \ \ (1 \ \ \ \ ) \ \hat{x}_i(1 \ \ \hat{x}_j) \text{ and } \mathbf{P}[i \in R, G] \ \ (1 \ \ \ \ ) \ \hat{x}_i, \ \forall i,j \in Q_z. \tag{37}$$

Combined with the above two inequalities, this would imply that conditioned on sample S,

$$\begin{aligned}
\mathsf{E}[ALG] \ \ (1 \ \ 2\ )(1 \ \ \ \ ) & \sum_{i \in Q_z} v_i r_i \hat{x}_i \sum_{j \in Q_z} \ _{ij}(1 \ \ \hat{x}_j) + \sum_{i \in Q_z} C_i(z)\hat{x}_i \\
(1 \ \ 2\ )(1 \ \ \ \ ) & \sum_{i \in Q_z} v_i r_i \hat{x}_i \ (\frac{i}{p} \ \ N\ ) + \sum_{i \in Q_z} C_i(z)\hat{x}_i \\
(1 \ \ 2\ )(1 \ \ \ \ ) & \ LP(\hat{\mathbf{x}}) \ \ N \sum_{i \in Q_z} v_i r_i .
\end{aligned}$$

where the second inequality follows from the feasibility of $\hat{\mathbf{x}}$ for $\mathrm{LP}_C(\ )$. Taking expectation over S and using Lemma 27,

$$\begin{aligned}
\mathsf{E}[ALG] \ \ (1 \ \ 2\ )(1 \ \ \ \ ) \ \ 1 \ \ \frac{3}{N^2} \ OPT \ \ 2N \sum_{i \in Q_z} v_i r_i & \\
(1 \ \ 7\ ) \ 1 \ \ \frac{3}{N^2} \ OPT \ \ \frac{24 \ OPT}{\ \ \ \ \ \ \ \ \ \ } \ \ 1 \ \ 10 \ \ \frac{24}{\ \ \ \ } \ \ OPT &
\end{aligned}$$

where the second inequality follows from Lemma 25 and $\ \ = 3$.

It only remains to prove (37). Note that

$$\mathbf{P}[i \in R, j \notin R, G] = \mathbf{P}[i \in R] \ \mathbf{P}[j \notin R] \ \mathbf{P}[G \mid i \in R, j \notin R] = \hat{x}_i(1 - \hat{x}_j) \ \mathbf{P}[G \mid i \in R, j \notin R]. \quad (38)$$

We now bound $\mathbf{P}[G \mid i \in R, j \notin R]$ using Chernoff bound. Using the definition of $\tilde{\mathbf{x}}$,

$$\mathbf{P}[G \mid i \in R, j \notin R] = \mathbf{P}\left[\sum_{t=1}^{} \tilde{x}_t \quad (1+ \ )c \mid \tilde{x}_i = 1, \tilde{x}_j = 0\right] = \mathbf{P}\left[\sum_{t \in [n] \setminus i} \tilde{x}_t \quad (1+ \ )c \quad 1\right]$$

Note that $= \mathsf{E}[\sum_{t \in [n] \setminus i} \tilde{x}_t] \quad c$ and $c \quad 2 \quad N > 4$ as $N \quad \frac{4}{}$. So,

$$\mathbf{P}\left[\sum_{t \in [n] \setminus i} \tilde{x}_t > (1+ \ )c \quad 1\right] \quad \mathbf{P}\left[\sum_{t \in [n] \setminus i} \tilde{x}_t > (1+\frac{}{2})c\right] \quad e^{\ 2c=12} \quad e^{\ 2c=2} \quad e^{\ 2 \ N=2} \quad ;$$

where the second inequality is by Chernoff bound and the last inequality uses $N \quad \frac{2}{3} \quad \frac{2}{2 \ \ln(1= \ )}$. Combining the above two inequalities, $\mathbf{P}[G \mid i \in R, j \notin R] \quad 1 \quad$. Together with (38) this proves the first inequality in (37). The second inequality in (37) follows identically (there is no node j).

This completes the proof of Theorem 14. For any $^{-} \in (0;1]$, using $= ^{-} =34$ and Theorem 14, we obtain an expected objective at least $(1 \quad ^{-}) \ OPT$. So this is a randomized PTAS for (34) when is a constant. Note that this provides a guarantee on the expected objective. We can obtain a high-probability guarantee in the exact same way as described in §6 for the unconstrained problem.

Finally, to complete the proof of Theorem 12, we use the high-probability guarantee with the binary-search based framework, again as described in §6 for the unconstrained problem.

Appendix F: Computational Results

In this section, we provide a comprehensive analysis of the computational results of our approximation algorithms for unconstrained, capacitated, 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 1, 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 $100 \quad (\mathbf{x}_{ALG})/\hat{z}$ where $\mathbf{x}_{ALG}$ is our algorithm's solution. This quantity gives us a lower bound (percentage) on how close our solution is to the optimal.

Basic 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 $r_i = 1 \quad v_i$ which captures the intuition of more expensive products being less preferable. Parameters I and C refer to independent and correlated instances respectively. The number of products n is 50 or 100. The dissimilarity parameters for the nests are sampled from the uniform distribution $[0,\bar{\ }]$ where $\bar{\ }$ is a parameter that is varied over $\{0.25, 0.5, 0.75\}$. For our experiments, we used symmetric parameters, i.e. $_{ij} = _{ji}$ for all $i,j \in [n]$. The value P_0 is used to generate the weight of the no-purchase option as follows: $v_0 = P_0(\sum_{(i,j)\in M} V_{ij}(\mathbf{1})\ ^{ij})/(1 \quad P_0)$, where $\mathbf{1}$ is a vector of all 1's. The value of P_0 is the probability that a customer selects the no-purchase option when all the products are offered. The parameter P_0 is varied over $\{0.25, 0.5, 0.75\}$. The test problems are labeled according to these parameter configurations. The test problems are labeled as $(T,n,\bar{\ },P_0) \in \{I,C\} \quad \{50,100\} \quad \{0.1,0.5,1.0\}$ $\{0.25,0.5,0.75\}$ where T is a placeholder for the type of the problem. In this way, 36 configurations are generated.

Reported quantities: For each configuration, we report the following. The first column states the parameter combination. The second column shows the algorithm's average performance over all 100 instances. The third column records the worst-case performance over the 100 instances. The fourth and fifth columns show the 5^{th} and 95^{th} percentile of the performance. The sixth column shows the standard deviation over the 100 instances. The last column gives the average computation time (in seconds).

Unconstrained assortment optimization: We only tested the LP-based 0.5-approximation algorithm that follows from the capacitated version. Table 2 shows that our approach performs very well. Even the worst-case guarantee does not drop below 99%. The time taken for execution is typically under 2 seconds.

Capacitated assortment optimization: The capacitated test problems are generated in the same way as the unconstrained problems. The only new parameter needed is the capacity constraint of the problem. The capacity $c = d\ ne$, where the parameter $\quad$ is varied over $\{0.2, 0.5, 0.8\}$. The test problems are labeled as $(T,n,\bar{\ },P_0,\ ) \in \{I,C\} \quad \{50,100\} \quad \{0.1,0.5,1.0\} \quad \{0.25,0.75\}$ $\{0.2,0.5,0.8\}$. In this way, 72 configurations are generated. The results are in Tables 3 and 4. Our approximation algorithm performs very well. The average stays close to 99% in most cases, and even the 5^{th} percentile does not fall below 90%. The time taken for execution is roughly the same as for the unconstrained case. Moreover, we observe that about 64% of the instances result in binding constraints with respect to the LP solution.

Knapsack constrained assortment optimization: The instances for assortment optimization with a knapsack constraint are generated in a similar fashion as the earlier problems. Since the

Param. Conf.	$(\hat{x})/\hat{z}$					CPU
$(T;n;\bar{};P_0)$	Avg.	Min.	5th	95th	Std.	Secs.
$(I;50;0:1;0:25)$	99.83	99.48	99.62	99.96	0.1	0.41
$(I;50;0:1;0:5)$	99.79	99.36	99.57	99.97	0.12	0.43
$(I;50;0:1;0:75)$	99.75	99.37	99.46	99.95	0.14	0.44
$(I;50;0:5;0:25)$	99.97	99.8	99.89	100.0	0.04	0.44
$(I;50;0:5;0:5)$	99.96	99.78	99.86	100.0	0.05	0.49
$(I;50;0:5;0:75)$	99.96	99.75	99.81	100.0	0.06	0.5
$(I;50;1:0;0:25)$	99.99	99.9	99.94	100.0	0.02	0.45
$(I;50;1:0;0:5)$	99.98	99.82	99.92	100.0	0.03	0.52
$(I;50;1:0;0:75)$	99.98	99.82	99.9	100.0	0.04	0.52
$(I;100;0:1;0:25)$	99.84	99.73	99.74	99.92	0.05	1.72
$(I;100;0:1;0:5)$	99.8	99.63	99.66	99.9	0.07	1.78
$(I;100;0:1;0:75)$	99.79	99.64	99.67	99.87	0.06	1.78
$(I;100;0:5;0:25)$	99.97	99.92	99.93	99.99	0.02	1.78
$(I;100;0:5;0:5)$	99.96	99.86	99.93	99.99	0.02	1.94
$(I;100;0:5;0:75)$	99.96	99.89	99.91	99.99	0.02	1.99
$(I;100;1:0;0:25)$	99.98	99.94	99.96	100.0	0.01	1.84
$(I;100;1:0;0:5)$	99.98	99.94	99.96	100.0	0.01	2.0
$(I;100;1:0;0:75)$	99.98	99.94	99.96	100.0	0.01	1.99

Param. Conf.	$(\hat{x})/\hat{z}$					CPU
$(T;n;\bar{};P_0)$	Avg.	Min.	5th	95th	Std.	Secs.
$(C;50;0:1;0:25)$	99.55	99.12	99.24	99.79	0.15	0.42
$(C;50;0:1;0:5)$	99.55	99.17	99.34	99.74	0.13	0.45
$(C;50;0:1;0:75)$	99.57	98.85	99.22	99.81	0.18	0.45
$(C;50;0:5;0:25)$	99.9	99.68	99.79	100.0	0.07	0.45
$(C;50;0:5;0:5)$	99.92	99.67	99.79	100.0	0.07	0.47
$(C;50;0:5;0:75)$	99.94	99.7	99.79	100.0	0.07	0.49
$(C;50;1:0;0:25)$	99.97	99.82	99.88	100.0	0.04	0.46
$(C;50;1:0;0:5)$	99.97	99.84	99.89	100.0	0.04	0.48
$(C;50;1:0;0:75)$	99.98	99.84	99.9	100.0	0.03	0.49
$(C;100;0:1;0:25)$	99.57	99.37	99.43	99.69	0.08	1.66
$(C;100;0:1;0:5)$	99.59	99.29	99.4	99.71	0.1	1.77
$(C;100;0:1;0:75)$	99.58	99.21	99.41	99.74	0.1	1.82
$(C;100;0:5;0:25)$	99.92	99.84	99.86	99.97	0.03	1.81
$(C;100;0:5;0:5)$	99.93	99.84	99.87	99.96	0.03	1.9
$(C;100;0:5;0:75)$	99.93	99.82	99.88	99.98	0.03	1.99
$(C;100;1:0;0:25)$	99.97	99.93	99.94	99.99	0.02	1.92
$(C;100;1:0;0:5)$	99.97	99.92	99.93	100.0	0.02	1.97
$(C;100;1:0;0:75)$	99.98	99.92	99.94	100.0	0.02	1.99

Table 2 Unconstrained Instances

sizes are robust to scaling, we set the capacity $c = 1$, and generate the sizes for the products uniformly in the range $[0; \ ]$ for $\ 2\ f0:1;0:25;0:5;1g$. The test problems are labeled as $(T;n;\bar{};P_0;\ )\ 2\ fI;Cg\ \ f50;100g\ \ f0:1;0:5;1:0g\ \ f0:25;0:75g\ \ f0:1;0:25;0:5;1:0g$. In this way, 96 configurations are generated. We use the $0:25$ approximation algorithm from Theorem 6. We chose to implement this algorithm instead of the stronger $(0:5\ \ )$-approximation algorithm (Theorem 8) because the latter one involves a significant enumeration step that is likely to be slow for our test instances.Our results for this version are in Tables 5 and 6. Again, our approximation algorithm performs very well. For most cases, the average stays above 95%, and the 5^{th} percentile is above 90%. The worst 5^{th} percentile performance is 77%. The execution time does not change from the previous cases, even though this is a more general problem. We would like to note that about 99% of the knapsack instances result in binding constraints with respect to the LP solution.

Param. Conf.	$(\hat{x})/\hat{z}$					CPU
$(I; 50; 0:1; 0:25; 0:8)$	99.84	99.47	99.65	99.96	0.09	0.41
$(I; 50; 0:1; 0:25; 0:5)$	99.83	99.5	99.66	99.96	0.09	0.41
$(I; 50; 0:1; 0:25; 0:2)$	99.79	99.36	99.5	100.0	0.14	0.39
$(I; 50; 0:1; 0:75; 0:8)$	99.75	99.39	99.51	99.95	0.14	0.45
$(I; 50; 0:1; 0:75; 0:5)$	99.7	99.16	99.34	99.94	0.17	0.44
$(I; 50; 0:1; 0:75; 0:2)$	99.7	99.01	99.18	100.0	0.25	0.43
$(I; 50; 0:5; 0:25; 0:8)$	99.97	99.69	99.91	100.0	0.04	0.44
$(I; 50; 0:5; 0:25; 0:5)$	99.96	99.81	99.88	100.0	0.04	0.44
$(I; 50; 0:5; 0:25; 0:2)$	99.97	99.79	99.88	100.0	0.05	0.42
$(I; 50; 0:5; 0:75; 0:8)$	99.96	99.75	99.87	100.0	0.05	0.52
$(I; 50; 0:5; 0:75; 0:5)$	99.93	99.63	99.76	100.0	0.08	0.53
$(I; 50; 0:5; 0:75; 0:2)$	99.95	99.68	99.77	100.0	0.09	0.51
$(I; 50; 1:0; 0:25; 0:8)$	99.99	99.9	99.94	100.0	0.02	0.46
$(I; 50; 1:0; 0:25; 0:5)$	99.98	99.87	99.92	100.0	0.03	0.46
$(I; 50; 1:0; 0:25; 0:2)$	99.98	99.77	99.88	100.0	0.04	0.43
$(I; 50; 1:0; 0:75; 0:8)$	99.98	99.75	99.89	100.0	0.04	0.52
$(I; 50; 1:0; 0:75; 0:5)$	99.97	99.74	99.83	100.0	0.05	0.52
$(I; 50; 1:0; 0:75; 0:2)$	99.98	99.77	99.81	100.0	0.06	0.5

Param. Conf.	$(\hat{x})/\hat{z}$					CPU
$(C; 50; 0:1; 0:25; 0:8)$	99.6	99.24	99.38	99.8	0.13	0.43
$(C; 50; 0:1; 0:25; 0:5)$	99.46	98.99	99.13	99.73	0.17	0.43
$(C; 50; 0:1; 0:25; 0:2)$	95.31	93.0	93.77	99.28	1.76	0.5
$(C; 50; 0:1; 0:75; 0:8)$	99.58	99.14	99.25	99.85	0.17	0.46
$(C; 50; 0:1; 0:75; 0:5)$	98.15	85.83	90.19	99.63	2.76	0.55
$(C; 50; 0:1; 0:75; 0:2)$	91.68	90.34	90.65	92.25	1.15	0.58
$(C; 50; 0:5; 0:25; 0:8)$	99.93	99.81	99.85	100.0	0.05	0.45
$(C; 50; 0:5; 0:25; 0:5)$	99.9	99.69	99.78	100.0	0.06	0.46
$(C; 50; 0:5; 0:25; 0:2)$	97.32	95.17	95.35	100.0	1.88	0.54
$(C; 50; 0:5; 0:75; 0:8)$	99.92	99.69	99.77	100.0	0.07	0.5
$(C; 50; 0:5; 0:75; 0:5)$	99.53	91.83	97.42	100.0	1.11	0.55
$(C; 50; 0:5; 0:75; 0:2)$	94.1	92.53	93.05	99.79	1.69	0.68
$(C; 50; 1:0; 0:25; 0:8)$	99.97	99.83	99.88	100.0	0.04	0.46
$(C; 50; 1:0; 0:25; 0:5)$	99.95	99.8	99.88	100.0	0.04	0.46
$(C; 50; 1:0; 0:25; 0:2)$	99.18	96.91	97.12	100.0	1.05	0.53
$(C; 50; 1:0; 0:75; 0:8)$	99.97	99.84	99.88	100.0	0.04	0.51
$(C; 50; 1:0; 0:75; 0:5)$	99.94	99.68	99.76	100.0	0.07	0.54
$(C; 50; 1:0; 0:75; 0:2)$	97.62	95.25	95.6	100.0	1.74	0.65

Table 3 Capacitated Instances (n=50)

Assortment optimization under partition constraints: We also test the quality of our binary-search based framework (Theorem 7) for the assortment optimization problem under general constraints. We use the local search procedure from Lee et al. (2010) which is a 0:25-approximation algorithm for max-dicut under partition constraints. Combined with the binary search framework, this is a 0:25-approximation algorithm for the assortment problem under partition constraints. We make one simple change to the local search algorithm: we begin each local search iteration (within the binary search framework) with the approximate solution obtained from the previous iteration. This led to a somewhat better running time. We did not test the improved 0:385-approximation algorithm (Theorem 9) as it is much more complex.

We use the same instance generation procedure as before. In addition, parameter $k \in \{3; 7\}$ denotes the number of parts. Products are assigned randomly to the parts. The parameter $\in \{0:4; 0:8\}$ restricts the capacity of each part: if there are p products in a part then its capacity is

Param. Conf.	$(\hat{x})/\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ )$	Avg.	Min.	5th	95th	Std.	Secs.
$(I;100;0.1;0.25;0.8)$	99.83	99.65	99.73	99.91	0.06	1.71
$(I;100;0.1;0.25;0.5)$	99.84	99.67	99.73	99.91	0.05	1.72
$(I;100;0.1;0.25;0.2)$	99.82	99.67	99.72	99.92	0.07	1.76
$(I;100;0.1;0.75;0.8)$	99.78	99.59	99.66	99.87	0.06	1.82
$(I;100;0.1;0.75;0.5)$	99.71	99.49	99.57	99.83	0.08	1.82
$(I;100;0.1;0.75;0.2)$	99.75	99.39	99.52	99.92	0.12	1.82
$(I;100;0.5;0.25;0.8)$	99.97	99.87	99.94	100.0	0.02	1.86
$(I;100;0.5;0.25;0.5)$	99.97	99.89	99.93	99.99	0.02	1.82
$(I;100;0.5;0.25;0.2)$	99.96	99.85	99.9	100.0	0.03	1.91
$(I;100;0.5;0.75;0.8)$	99.95	99.89	99.91	99.99	0.03	2.06
$(I;100;0.5;0.75;0.5)$	99.94	99.82	99.89	99.99	0.03	2.04
$(I;100;0.5;0.75;0.2)$	99.95	99.82	99.85	100.0	0.05	1.98
$(I;100;1.0;0.25;0.8)$	99.98	99.94	99.96	100.0	0.01	1.87
$(I;100;1.0;0.25;0.5)$	99.98	99.93	99.94	100.0	0.02	1.88
$(I;100;1.0;0.25;0.2)$	99.98	99.91	99.94	100.0	0.02	1.9
$(I;100;1.0;0.75;0.8)$	99.98	99.93	99.95	100.0	0.02	2.01
$(I;100;1.0;0.75;0.5)$	99.98	99.91	99.94	100.0	0.02	2.09
$(I;100;1.0;0.75;0.2)$	99.98	99.79	99.9	100.0	0.04	2.0

Param. Conf.	$(\hat{x})/\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ )$	Avg.	Min.	5th	95th	Std.	Secs.
$(C;100;0.1;0.25;0.8)$	99.57	99.38	99.41	99.68	0.08	1.68
$(C;100;0.1;0.25;0.5)$	99.47	99.29	99.33	99.59	0.08	1.68
$(C;100;0.1;0.25;0.2)$	94.54	93.28	93.37	99.26	1.76	2.03
$(C;100;0.1;0.75;0.8)$	99.61	99.42	99.46	99.77	0.09	1.83
$(C;100;0.1;0.75;0.5)$	98.77	92.09	94.81	99.52	1.32	2.23
$(C;100;0.1;0.75;0.2)$	90.84	90.22	90.47	91.25	0.24	2.1
$(C;100;0.5;0.25;0.8)$	99.92	99.84	99.86	99.97	0.03	1.81
$(C;100;0.5;0.25;0.5)$	99.9	99.8	99.83	99.96	0.04	1.84
$(C;100;0.5;0.25;0.2)$	96.45	94.98	95.07	99.9	1.88	2.16
$(C;100;0.5;0.75;0.8)$	99.93	99.85	99.88	99.98	0.03	2.0
$(C;100;0.5;0.75;0.5)$	99.71	96.28	98.77	99.95	0.6	2.13
$(C;100;0.5;0.75;0.2)$	92.97	92.6	92.69	93.22	0.4	2.3
$(C;100;1.0;0.25;0.8)$	99.97	99.92	99.93	100.0	0.02	1.93
$(C;100;1.0;0.25;0.5)$	99.96	99.9	99.92	99.99	0.02	1.88
$(C;100;1.0;0.25;0.2)$	99.16	96.64	96.85	100.0	1.09	2.07
$(C;100;1.0;0.75;0.8)$	99.97	99.91	99.94	100.0	0.02	2.08
$(C;100;1.0;0.75;0.5)$	99.94	99.47	99.88	100.0	0.06	2.11
$(C;100;1.0;0.75;0.2)$	96.63	95.12	95.24	99.95	1.75	2.34

Table 4 Capacitated Instances (n=100)

set to p. The test problems are labeled as $(T;n;\bar{\ };P_0;\ ;k)$ 2 $fI;Cg$ $f50;100g$ $f0.1;0.5;1.0g$ $f0.25;0.75g$ $f0.4;0.8g$ $f3;7g$. In this way, 96 configurations are generated. We give our results in Tables 7 and 8. For almost all the cases, the average stays above 99%. The worst 5^{th} percentile performance is 94%, while the worst-case performance is 90% for all cases. We see an increase in the execution time from about 2 seconds (for $n = 50$) to about 12 seconds (for $n = 100$). Still, the time requirement remains small.

Param. Conf.	$(\hat{\mathbf{x}})=\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ )$	Avg.	Min.	5th	95th	Std.	Secs.
$(I;50;0{:}1;0{:}25;0{:}1)$	99.72	99.13	99.27	99.93	0.18	0.4
$(I;50;0{:}1;0{:}25;0{:}25)$	97.82	92.83	94.68	99.72	1.56	0.39
$(I;50;0{:}1;0{:}25;0{:}5)$	95.19	82.88	87.53	99.33	3.57	0.39
$(I;50;0{:}1;0{:}25;1{:}0)$	93.37	74.3	79.58	99.27	5.59	0.39
$(I;50;0{:}1;0{:}75;0{:}1)$	98.94	96.82	97.6	99.67	0.63	0.44
$(I;50;0{:}1;0{:}75;0{:}25)$	96.64	90.52	91.91	99.29	2.25	0.44
$(I;50;0{:}1;0{:}75;0{:}5)$	94.13	77.3	85.86	99.28	4.41	0.44
$(I;50;0{:}1;0{:}75;1{:}0)$	90.67	62.66	78.21	99.02	6.88	0.44
$(I;50;0{:}5;0{:}25;0{:}1)$	99.83	99.02	99.48	100.0	0.18	0.43
$(I;50;0{:}5;0{:}25;0{:}25)$	98.35	93.25	94.9	99.84	1.42	0.43
$(I;50;0{:}5;0{:}25;0{:}5)$	95.48	85.21	87.44	99.75	3.86	0.42
$(I;50;0{:}5;0{:}25;1{:}0)$	93.49	73.41	81.9	99.68	5.89	0.43
$(I;50;0{:}5;0{:}75;0{:}1)$	99.26	97.64	98.16	99.87	0.51	0.52
$(I;50;0{:}5;0{:}75;0{:}25)$	96.14	88.39	91.1	99.16	2.5	0.5
$(I;50;0{:}5;0{:}75;0{:}5)$	94.51	83.42	84.24	99.54	4.28	0.5
$(I;50;0{:}5;0{:}75;1{:}0)$	91.6	67.21	77.65	99.39	6.78	0.5
$(I;50;1{:}0;0{:}25;0{:}1)$	99.78	98.62	99.26	100.0	0.25	0.44
$(I;50;1{:}0;0{:}25;0{:}25)$	98.11	93.48	95.69	99.88	1.46	0.43
$(I;50;1{:}0;0{:}25;0{:}5)$	95.69	83.49	87.5	99.69	3.76	0.45
$(I;50;1{:}0;0{:}25;1{:}0)$	92.18	58.85	79.05	98.9	6.98	0.46
$(I;50;1{:}0;0{:}75;0{:}1)$	99.13	95.59	97.79	99.96	0.74	0.54
$(I;50;1{:}0;0{:}75;0{:}25)$	96.26	90.21	90.69	99.59	2.6	0.52
$(I;50;1{:}0;0{:}75;0{:}5)$	94.06	82.46	85.72	99.38	4.27	0.52
$(I;50;1{:}0;0{:}75;1{:}0)$	90.95	66.82	77.05	98.92	6.93	0.5

Param. Conf.	$(\hat{\mathbf{x}})=\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ )$	Avg.	Min.	5th	95th	Std.	Secs.
$(C;50;0{:}1;0{:}25;0{:}1)$	98.75	96.71	97.66	99.48	0.63	0.46
$(C;50;0{:}1;0{:}25;0{:}25)$	97.54	94.58	95.08	99.32	1.28	0.46
$(C;50;0{:}1;0{:}25;0{:}5)$	96.7	89.82	93.24	99.23	1.87	0.46
$(C;50;0{:}1;0{:}25;1{:}0)$	95.72	87.15	89.71	99.04	2.82	0.44
$(C;50;0{:}1;0{:}75;0{:}1)$	98.02	94.88	96.4	99.29	0.94	0.52
$(C;50;0{:}1;0{:}75;0{:}25)$	97.01	92.31	94.04	98.99	1.49	0.5
$(C;50;0{:}1;0{:}75;0{:}5)$	95.5	89.33	90.29	98.8	2.7	0.49
$(C;50;0{:}1;0{:}75;1{:}0)$	93.72	83.51	86.2	98.54	3.68	0.49
$(C;50;0{:}5;0{:}25;0{:}1)$	99.26	97.56	98.26	99.87	0.51	0.47
$(C;50;0{:}5;0{:}25;0{:}25)$	97.98	94.76	95.24	99.66	1.31	0.48
$(C;50;0{:}5;0{:}25;0{:}5)$	97.06	92.48	93.52	99.51	1.77	0.48
$(C;50;0{:}5;0{:}25;1{:}0)$	94.95	81.71	87.78	99.52	3.7	0.49
$(C;50;0{:}5;0{:}75;0{:}1)$	98.5	95.98	96.99	99.68	0.84	0.57
$(C;50;0{:}5;0{:}75;0{:}25)$	97.44	93.46	94.52	99.63	1.57	0.56
$(C;50;0{:}5;0{:}75;0{:}5)$	95.96	89.91	91.54	99.32	2.44	0.54
$(C;50;0{:}5;0{:}75;1{:}0)$	94.11	81.99	86.7	99.06	3.9	0.54
$(C;50;1{:}0;0{:}25;0{:}1)$	99.2	97.77	98.1	99.9	0.54	0.48
$(C;50;1{:}0;0{:}25;0{:}25)$	98.03	94.67	95.37	99.71	1.26	0.48
$(C;50;1{:}0;0{:}25;0{:}5)$	97.13	88.55	93.4	99.7	2.02	0.48
$(C;50;1{:}0;0{:}25;1{:}0)$	95.35	82.88	88.93	99.68	3.25	0.48
$(C;50;1{:}0;0{:}75;0{:}1)$	98.46	96.22	96.7	99.79	1.02	0.56
$(C;50;1{:}0;0{:}75;0{:}25)$	97.38	91.83	94.18	99.72	1.74	0.55
$(C;50;1{:}0;0{:}75;0{:}5)$	96.15	91.11	91.63	99.4	2.41	0.54
$(C;50;1{:}0;0{:}75;1{:}0)$	95.0	83.76	88.26	99.56	3.63	0.53

Table 5 Knapsack-constrained Instances (n=50)

Param. Conf.	$(\hat{\mathbf{x}})=\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ )$	Avg.	Min.	5th	95th	Std.	Secs.
$(I;100;0:1;0:25;0:1)$	99.24	97.92	98.53	99.76	0.42	1.76
$(I;100;0:1;0:25;0:25)$	97.65	92.54	94.7	99.7	1.57	1.75
$(I;100;0:1;0:25;0:5)$	96.21	88.94	91.4	99.33	2.67	1.72
$(I;100;0:1;0:25;1:0)$	94.62	83.33	84.92	99.17	4.05	1.76
$(I;100;0:1;0:75;0:1)$	98.63	95.72	97.03	99.58	0.82	1.89
$(I;100;0:1;0:75;0:25)$	96.98	89.85	92.83	99.49	2.24	1.85
$(I;100;0:1;0:75;0:5)$	95.9	88.17	90.07	99.44	2.83	1.85
$(I;100;0:1;0:75;1:0)$	93.08	78.69	83.72	99.11	5.0	1.85
$(I;100;0:5;0:25;0:1)$	99.4	98.03	98.55	99.91	0.44	1.82
$(I;100;0:5;0:25;0:25)$	98.12	95.1	95.34	99.83	1.34	1.82
$(I;100;0:5;0:25;0:5)$	96.47	89.09	91.83	99.55	2.47	1.82
$(I;100;0:5;0:25;1:0)$	94.25	77.52	83.6	99.16	4.85	1.8
$(I;100;0:5;0:75;0:1)$	98.89	96.06	97.3	99.79	0.84	1.99
$(I;100;0:5;0:75;0:25)$	97.17	91.01	93.66	99.68	1.97	1.96
$(I;100;0:5;0:75;0:5)$	96.02	86.46	90.26	99.39	2.86	1.94
$(I;100;0:5;0:75;1:0)$	93.73	74.69	83.14	99.0	5.06	1.92
$(I;100;1:0;0:25;0:1)$	99.36	97.75	98.54	99.93	0.46	1.85
$(I;100;1:0;0:25;0:25)$	98.0	93.43	95.64	99.69	1.41	1.84
$(I;100;1:0;0:25;0:5)$	96.27	88.04	90.97	99.53	2.67	1.89
$(I;100;1:0;0:25;1:0)$	95.17	84.25	86.21	99.75	4.03	1.89
$(I;100;1:0;0:75;0:1)$	98.77	96.19	97.06	99.84	0.86	2.01
$(I;100;1:0;0:75;0:25)$	97.17	92.02	93.35	99.66	1.91	2.01
$(I;100;1:0;0:75;0:5)$	96.02	86.68	89.74	99.75	3.0	1.97
$(I;100;1:0;0:75;1:0)$	92.93	77.25	83.15	99.45	5.1	1.97

Param. Conf.	$(\hat{\mathbf{x}})=\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ )$	Avg.	Min.	5th	95th	Std.	Secs.
$(C;100;0:1;0:25;0:1)$	98.7	97.35	97.81	99.27	0.48	1.73
$(C;100;0:1;0:25;0:25)$	97.92	96.07	96.52	99.14	0.84	1.79
$(C;100;0:1;0:25;0:5)$	97.01	93.23	94.48	99.09	1.48	1.78
$(C;100;0:1;0:25;1:0)$	96.09	87.37	92.0	98.97	2.17	1.75
$(C;100;0:1;0:75;0:1)$	98.31	96.99	97.23	99.21	0.63	1.9
$(C;100;0:1;0:75;0:25)$	97.12	93.9	95.27	98.92	1.17	1.85
$(C;100;0:1;0:75;0:5)$	96.33	91.39	92.64	98.77	1.84	1.83
$(C;100;0:1;0:75;1:0)$	95.07	87.95	89.05	98.39	2.76	1.81
$(C;100;0:5;0:25;0:1)$	99.22	98.24	98.39	99.79	0.44	1.87
$(C;100;0:5;0:25;0:25)$	98.42	96.4	96.8	99.68	0.88	1.93
$(C;100;0:5;0:25;0:5)$	97.8	93.05	94.7	99.57	1.44	1.91
$(C;100;0:5;0:25;1:0)$	96.22	89.65	91.41	99.59	2.42	1.88
$(C;100;0:5;0:75;0:1)$	98.86	97.09	97.52	99.76	0.69	2.07
$(C;100;0:5;0:75;0:25)$	98.06	95.85	96.23	99.63	1.11	1.99
$(C;100;0:5;0:75;0:5)$	97.03	92.06	93.77	99.61	1.94	1.97
$(C;100;0:5;0:75;1:0)$	95.65	89.21	90.26	99.3	2.7	1.93
$(C;100;1:0;0:25;0:1)$	99.14	98.13	98.37	99.78	0.44	1.94
$(C;100;1:0;0:25;0:25)$	98.52	96.34	97.16	99.72	0.84	1.96
$(C;100;1:0;0:25;0:5)$	97.6	93.6	94.69	99.73	1.6	1.93
$(C;100;1:0;0:25;1:0)$	96.13	89.22	92.25	99.45	2.34	1.89
$(C;100;1:0;0:75;0:1)$	98.7	97.33	97.53	99.8	0.74	2.1
$(C;100;1:0;0:75;0:25)$	98.07	95.51	95.9	99.81	1.2	2.01
$(C;100;1:0;0:75;0:5)$	96.96	92.97	93.86	99.4	1.73	1.96
$(C;100;1:0;0:75;1:0)$	95.68	86.95	90.18	99.63	2.84	1.95

Table 6 Knapsack-constrained Instances (n=100)

Param. Conf.	$(\hat{x})=\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ ;k)$	Avg.	Min.	5th	95th	Std.	Secs.
$(I;50;0.1;0.25;0.4;3)$	99.81	99.52	99.61	99.94	0.1	1.9
$(I;50;0.1;0.25;0.4;7)$	99.81	99.14	99.59	99.97	0.13	1.53
$(I;50;0.1;0.25;0.8;3)$	99.83	99.46	99.67	99.95	0.09	1.93
$(I;50;0.1;0.25;0.8;7)$	99.84	99.53	99.69	99.96	0.08	2.14
$(I;50;0.1;0.75;0.4;3)$	99.46	95.37	98.11	99.94	0.63	1.43
$(I;50;0.1;0.75;0.4;7)$	99.5	96.97	98.28	99.9	0.47	1.22
$(I;50;0.1;0.75;0.8;3)$	99.2	88.62	95.99	99.93	1.62	1.35
$(I;50;0.1;0.75;0.8;7)$	99.23	89.74	96.55	99.94	1.34	1.52
$(I;50;0.5;0.25;0.4;3)$	99.97	99.74	99.89	100.0	0.04	2.44
$(I;50;0.5;0.25;0.4;7)$	99.97	99.74	99.87	100.0	0.05	1.84
$(I;50;0.5;0.25;0.8;3)$	99.97	99.78	99.89	100.0	0.04	2.13
$(I;50;0.5;0.25;0.8;7)$	99.96	99.79	99.88	100.0	0.04	2.29
$(I;50;0.5;0.75;0.4;3)$	99.76	97.28	98.65	100.0	0.49	1.66
$(I;50;0.5;0.75;0.4;7)$	99.61	92.63	98.01	100.0	0.93	1.42
$(I;50;0.5;0.75;0.8;3)$	99.47	91.74	96.72	100.0	1.33	1.41
$(I;50;0.5;0.75;0.8;7)$	99.47	90.79	96.51	100.0	1.31	1.47
$(I;50;1.0;0.25;0.4;3)$	99.98	99.88	99.92	100.0	0.03	2.54
$(I;50;1.0;0.25;0.4;7)$	99.98	99.83	99.89	100.0	0.04	2.01
$(I;50;1.0;0.25;0.8;3)$	99.98	99.92	99.93	100.0	0.02	2.4
$(I;50;1.0;0.25;0.8;7)$	99.98	99.89	99.91	100.0	0.03	2.49
$(I;50;1.0;0.75;0.4;3)$	99.74	95.98	98.23	100.0	0.6	1.58
$(I;50;1.0;0.75;0.4;7)$	99.69	95.61	98.56	100.0	0.77	1.23
$(I;50;1.0;0.75;0.8;3)$	99.42	92.34	95.82	100.0	1.36	1.22
$(I;50;1.0;0.75;0.8;7)$	99.45	86.69	96.41	100.0	1.57	1.31

Param. Conf.	$(\hat{x})=\hat{z}$					CPU
$(T;n;\bar{\ };P_0;\ ;k)$	Avg.	Min.	5th	95th	Std.	Secs.
$(C;50;0.1;0.25;0.4;3)$	99.31	97.81	98.67	99.7	0.31	2.08
$(C;50;0.1;0.25;0.4;7)$	99.29	98.38	98.54	99.72	0.34	1.68
$(C;50;0.1;0.25;0.8;3)$	99.57	99.13	99.3	99.78	0.15	1.77
$(C;50;0.1;0.25;0.8;7)$	99.55	99.13	99.28	99.82	0.15	2.04
$(C;50;0.1;0.75;0.4;3)$	96.7	90.7	92.45	99.43	2.14	1.11
$(C;50;0.1;0.75;0.4;7)$	96.47	92.24	92.78	98.82	1.71	0.86
$(C;50;0.1;0.75;0.8;3)$	99.42	98.09	98.87	99.75	0.3	0.84
$(C;50;0.1;0.75;0.8;7)$	99.35	98.34	98.72	99.82	0.34	0.74
$(C;50;0.5;0.25;0.4;3)$	99.86	99.43	99.62	100.0	0.12	2.21
$(C;50;0.5;0.25;0.4;7)$	99.84	99.18	99.58	100.0	0.13	1.75
$(C;50;0.5;0.25;0.8;3)$	99.93	99.74	99.82	100.0	0.06	1.72
$(C;50;0.5;0.25;0.8;7)$	99.92	99.67	99.79	100.0	0.06	1.96
$(C;50;0.5;0.75;0.4;3)$	98.76	93.88	95.65	100.0	1.25	1.03
$(C;50;0.5;0.75;0.4;7)$	98.25	93.81	95.48	99.86	1.37	0.8
$(C;50;0.5;0.75;0.8;3)$	99.82	98.15	99.48	100.0	0.24	0.84
$(C;50;0.5;0.75;0.8;7)$	99.76	98.64	99.02	100.0	0.28	0.92
$(C;50;1.0;0.25;0.4;3)$	99.93	99.56	99.8	100.0	0.08	2.22
$(C;50;1.0;0.25;0.4;7)$	99.94	99.77	99.8	100.0	0.06	1.84
$(C;50;1.0;0.25;0.8;3)$	99.96	99.81	99.89	100.0	0.04	1.72
$(C;50;1.0;0.25;0.8;7)$	99.96	99.83	99.88	100.0	0.04	1.86
$(C;50;1.0;0.75;0.4;3)$	99.68	97.58	98.89	100.0	0.41	1.09
$(C;50;1.0;0.75;0.4;7)$	99.52	97.88	98.31	100.0	0.51	0.83
$(C;50;1.0;0.75;0.8;3)$	99.85	98.11	99.45	100.0	0.28	0.86
$(C;50;1.0;0.75;0.8;7)$	99.84	98.56	99.24	100.0	0.27	0.81

Table 7 Partition Instances (n=50)

Param. Conf.	$(\hat{x})=\hat{z}$					CPU
$(I;100;0.1;0.25;0.4;3)$	99.82	99.64	99.68	99.9	0.06	14.08
$(I;100;0.1;0.25;0.4;7)$	99.81	99.65	99.73	99.9	0.05	10.12
$(I;100;0.1;0.25;0.8;3)$	99.84	99.67	99.77	99.91	0.04	13.38
$(I;100;0.1;0.25;0.8;7)$	99.84	99.71	99.77	99.92	0.05	13.93
$(I;100;0.1;0.75;0.4;3)$	99.55	96.57	98.76	99.81	0.42	9.91
$(I;100;0.1;0.75;0.4;7)$	99.49	95.92	98.75	99.81	0.49	6.76
$(I;100;0.1;0.75;0.8;3)$	99.31	90.65	97.44	99.86	1.42	6.96
$(I;100;0.1;0.75;0.8;7)$	99.38	91.86	97.53	99.88	1.08	6.57
$(I;100;0.5;0.25;0.4;3)$	99.96	99.9	99.93	100.0	0.02	14.06
$(I;100;0.5;0.25;0.4;7)$	99.97	99.9	99.92	99.99	0.02	10.97
$(I;100;0.5;0.25;0.8;3)$	99.97	99.9	99.93	99.99	0.02	13.43
$(I;100;0.5;0.25;0.8;7)$	99.97	99.91	99.93	99.99	0.02	13.65
$(I;100;0.5;0.75;0.4;3)$	99.69	96.69	97.71	99.98	0.67	9.45
$(I;100;0.5;0.75;0.4;7)$	99.73	96.38	99.06	99.97	0.52	6.87
$(I;100;0.5;0.75;0.8;3)$	99.29	92.11	95.41	99.98	1.41	6.32
$(I;100;0.5;0.75;0.8;7)$	99.63	91.46	97.7	99.99	1.09	7.42
$(I;100;1.0;0.25;0.4;3)$	99.98	99.91	99.96	100.0	0.02	14.37
$(I;100;1.0;0.25;0.4;7)$	99.98	99.94	99.96	100.0	0.01	10.65
$(I;100;1.0;0.25;0.8;3)$	99.98	99.39	99.97	100.0	0.06	13.18
$(I;100;1.0;0.25;0.8;7)$	99.99	99.94	99.96	100.0	0.01	14.08
$(I;100;1.0;0.75;0.4;3)$	99.75	97.16	98.54	100.0	0.55	9.07
$(I;100;1.0;0.75;0.4;7)$	99.67	93.37	98.26	100.0	0.97	7.14
$(I;100;1.0;0.75;0.8;3)$	99.51	90.22	97.82	100.0	1.44	7.12
$(I;100;1.0;0.75;0.8;7)$	99.52	94.01	95.05	100.0	1.25	6.94

Param. Conf.	$(\hat{x})=\hat{z}$					CPU
$(C;100;0.1;0.25;0.4;3)$	99.38	99.04	99.17	99.56	0.11	15.01
$(C;100;0.1;0.25;0.4;7)$	99.34	98.66	99.13	99.54	0.15	10.83
$(C;100;0.1;0.25;0.8;3)$	99.57	99.32	99.39	99.7	0.09	11.34
$(C;100;0.1;0.25;0.8;7)$	99.57	99.32	99.42	99.7	0.09	12.39
$(C;100;0.1;0.75;0.4;3)$	96.94	90.6	93.68	99.19	1.8	6.08
$(C;100;0.1;0.75;0.4;7)$	96.71	92.18	94.0	98.95	1.49	3.9
$(C;100;0.1;0.75;0.8;3)$	99.45	98.63	99.0	99.68	0.21	4.45
$(C;100;0.1;0.75;0.8;7)$	99.41	98.11	98.94	99.69	0.27	4.63
$(C;100;0.5;0.25;0.4;3)$	99.89	99.8	99.81	99.95	0.04	14.4
$(C;100;0.5;0.25;0.4;7)$	99.85	98.67	99.68	99.95	0.14	11.22
$(C;100;0.5;0.25;0.8;3)$	99.92	99.76	99.87	99.97	0.03	11.94
$(C;100;0.5;0.25;0.8;7)$	99.92	99.85	99.87	99.96	0.03	12.73
$(C;100;0.5;0.75;0.4;3)$	99.4	97.19	98.11	99.92	0.55	8.36
$(C;100;0.5;0.75;0.4;7)$	98.97	96.33	96.85	99.85	0.87	4.96
$(C;100;0.5;0.75;0.8;3)$	99.83	99.14	99.39	99.97	0.18	6.04
$(C;100;0.5;0.75;0.8;7)$	99.84	99.18	99.54	99.97	0.15	5.86
$(C;100;1.0;0.25;0.4;3)$	99.95	99.86	99.88	99.99	0.03	14.68
$(C;100;1.0;0.25;0.4;7)$	99.94	99.82	99.88	99.98	0.03	10.55
$(C;100;1.0;0.25;0.8;3)$	99.97	99.91	99.94	99.99	0.02	10.73
$(C;100;1.0;0.25;0.8;7)$	99.97	99.91	99.93	99.99	0.02	11.14
$(C;100;1.0;0.75;0.4;3)$	99.69	98.65	98.99	99.97	0.29	5.53
$(C;100;1.0;0.75;0.4;7)$	99.67	98.75	99.05	99.97	0.3	4.42
$(C;100;1.0;0.75;0.8;3)$	99.85	99.0	99.46	99.99	0.19	4.72
$(C;100;1.0;0.75;0.8;7)$	99.85	98.96	99.37	100.0	0.21	3.87

Table 8 Partition Instances (n=100)