Fair Assortment Planning

Qinyi Chen
Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, qinyic@mit.edu

Negin Golrezaei
Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, golrezae@mit.edu

Fransisca Susan
Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, fsusan@mit.edu

Edy Baskoro
Combinatorial Mathematics Research Group, Institut Teknologi Bandung, Bandung, Indonesia, ebaskoro@gmail.com

Many online platforms, ranging from online retail stores to social media platforms, employ algorithms to optimize their offered assortment of items (e.g., products and contents). These algorithms tend to prioritize the platforms’ short-term goals by featuring items with the highest popularity. However, this practice can then lead to too little visibility for the rest of the items, making them leave the platform, and in turn hurting the platform’s long-term goals. Motivated by that, we introduce and study a fair assortment planning problem, which requires any two items with similar merits (popularities) to be offered similar visibility. We show that the problem can be formulated as a linear program (LP), called (fair), that optimizes over the distribution of all feasible assortments. To find a near-optimal solution to (fair), we propose a framework based on the Ellipsoid method, which requires a polynomial-time separation oracle to the dual of the LP. We show that finding an optimal separation oracle to the dual problem is an NP-complete problem, and hence we propose two approximate separation oracles: a 1/2-approx. algorithm and an FPTAS. The approximate separation oracles result in a polynomial-time 1/2-approx. algorithm and an FPTAS for the original problem (fair) using the Ellipsoid method. Further, they are designed by (i) showing the separation oracle to the dual of the LP is equivalent to solving an infinite series of parameterized knapsack problems, and (ii) taking advantage of the structure of the parameterized knapsack problems. Finally, we conduct a case study using the MovieLens dataset, demonstrating the efficacy of our 1/2-approx. and FPTAS fair Ellipsoid-based assortment planning algorithms.

Key words: assortment planning, fairness, online platforms, approximation algorithm, customer exposure

1. Introduction

Algorithms have been extensively used by modern online platforms, such as online retail stores and social media platforms, to optimize their operational decisions, ranging from assortment planning (e.g., Kok et al. (2008), Rusmevichientong et al. (2010), Davis et al. (2014), Golrezaei et al. (2014)) to pricing (e.g., Derakhshan et al. (2019), Cohen et al. (2020), Golrezaei et al. (2021a)) to product ranking (e.g., Derakhshan et al. (2020), Golrezaei et al. (2021b), Niazadeh et al. (2021)). While the use of algorithms can make the decision-making processes more efficient, they can also create
unfair environments for some of the crucial players of the online platforms. In assortment planning, for example, *single-minded* algorithms that are only concerned about the platforms’ short-term objective (e.g., revenue or market share) can treat some of the items (e.g., products on online retail stores; posts on social media) on the platforms unfairly. Such algorithms usually solely prioritize the most lucrative or popular items in the offered assortment, leading to insignificant visibility for the rest of the items. The lack of visibility can then motivate the unprioritized items to leave the platform, creating unhealthy ecosystem which can have negative long-term consequences for the platform. This raises the following research question: *How can we design fair assortment planning algorithms under which any two items with similar merits get similar visibility?*

As alluded earlier, answering the question above can be of interest to a number of online platforms, such as online retail stores, social media platforms, job search sites, movie/music recommendation sites, to name a few. In all of these online platforms, a “winner-take-all” phenomenon can become prevalent due to the wide adoption of algorithmic recommendations:

- In online retail stores like Amazon, the platform often features the most popular products on their front page (e.g., as “Amazon’s Choices”), while many products of similar ratings and prices are displayed in subsequent pages. It then becomes increasingly difficult for these less popular products to obtain visibility in the highly competitive marketplace, as most Amazon customers rarely scroll past the first page of search results (Search Engine Journal 2018).

- Social media platforms like Instagram is widely adopted by business owners for marketing purposes. However, the recommendation algorithms used by Instagram tend to prioritize high-quality social media posts made with special camera effects, while many small businesses lack necessary video-editing equipment and skills. This then leads to an unfair outcome that small business owners have seen a notable decrease in the amount of user engagement on their posts, as well as in the number of their sales (The New York Times 2022).

- On job search sites such as LinkedIn, the items correspond to users who network with each other and search for job openings. By design of the algorithm, active users tend to be connected to other users or job postings more frequently, due to their greater representation in the data collected (VentureBeat 2021). However, this creates an unfair environment where the most active users keep expanding their network, while the less active users get very little visibility among other users or recruiters.

- Finally, the algorithms used by many movie/music recommendation sites are self-reinforcing and the cultural products they recommend to a specific user are often restricted to a few genres or types (Vox 2019). This raises the concern that certain cultural products might not survive the algorithms since they do not get sufficient visibility among the audience. Consequently, culture can become more similar than different.
All of the examples above demonstrate the practical importance of taking fairness into account while performing assortment planning for different types of items on a variety of online platforms.

To address the research question, we consider an online platform that hosts $n$ items, which can represent products, social media posts, job candidates, movies/music in corresponding contexts. Each item $i$ has a popularity weight $w_i > 0$ that is a proxy for the item $i$’s merit. The platform’s goal is to maximize its market share by optimizing over its offered assortment (with a cardinality of at most $K$) to its users who make a choice according to the multinomial choice model with popularity weight $w_i$, $i \in [n]$. Here, market share can be interpreted differently in respective online platforms. For online marketplaces, the market share is the probability of purchase of the offered assortment; for job search sites, the market share refers to the likelihood of establishing connections; for social media platforms, the market share measures the amount of user engagement on the featured posts. Further, on platforms such as online marketplaces, the item $i$ also has a profit margin $r_i > 0$ in addition to its popularity weight, which is a monetary value that the platform earns when the user chooses/purchases this item. For such kinds of online platforms, we can consider an alternative platform’s goal, which is to maximize its revenue.

For any of the online platforms discussed above, we would like to provide a fair environment for each item. To do so, we consider a parameterized notion of fairness that is based on parity of pairwise visibility across items. In this notion, items with similar popularity weights are expected to enjoy similar visibility, where an item’s visibility is the probability that the item is offered in the assortment; see Section 3 for the formal definition of this notion.\(^1\) We note that pairwise notions of fairness are quite common in the literature and have been used in various settings, including recommendation systems (Beutel et al. 2019), ranking and regression models (Narasimhan et al. 2020, Kuhlman et al. 2019, Singh et al. 2021), and predictive risk scores (Kallus and Zhou 2019).

1.1. Our Contributions

**Fair assortment planning problem.** We introduce a novel pairwise fairness notion to study fair assortment planning problems. While assortment planning problems with different objective functions have been widely studied in the literature (see Kok et al. (2008) for a survey), prior to this work, enforcing fairness in assortment planning problems has not been studied in the literature. In Section 3, we show that the fair assortment planning problem can be formulated as a Linear Program (LP) with exponentially many variables and $O(n^2)$ pairwise fairness constraints. We further show, in Section 4.1, that there exists an optimal solution to the aforementioned optimization problem that involves randomization over at most $O(n^2)$ assortments (see Theorem 1), making the implementation of the optimal solution practically feasible.

\(^1\) A more general notion of fairness can also be adopted, which enforces fairness with respect to any quality function of items (see discussion in Section 3).
Fair Ellipsoid-based assortment planning algorithms. In Section 4, we present a framework that provides a wide range of near-optimal algorithms for the fair assortment planning problem that we denote by Problem (FAIR). Our framework uses the Ellipsoid method and a near-optimal polynomial-time separation oracle to the dual counterpart of the fair assortment planning problem, denoted by Problem (FAIR-DUAL). Ideally, the Ellipsoid method would require an optimal separation oracle for Problem (FAIR-DUAL), where the separation oracle would either declare a point is feasible or return a hyperplane that separates the point from the convex set that describes the feasible region. Unfortunately, we show that designing an optimal separation oracle for Problem (FAIR-DUAL) is an NP-complete problem (see Theorem 2). However, we show that by designing a \( \beta \)-approximate separation oracle for Problem (FAIR-DUAL), with the help of the Ellipsoid method, we can design a \( \beta \)-approximate solution to Problem (FAIR), where \( \beta \in (0,1) \). See Theorem 3.

Separation oracle as solving an infinite number of knapsack problems. Having established this result (i.e., Theorem 3), we then proceed to design near-optimal separation oracles for Problem (FAIR-DUAL). To do so, we show that we need to find a set \( S \) with a cardinality of at most \( K \) that maximizes a cost-adjusted market share, where the costs are functions of dual variables associated with the fairness constraints. To solve this problem, we take advantage of a novel transformation that shows maximizing cost-adjusted market share is equivalent to solving an infinite series of knapsack problems with both cardinality and capacity constraints (see Theorem 4). Each knapsack problem is parameterized by its capacity \( W \in (0, \infty) \), where parameter \( W \) also influences the utility of items: the larger the \( W \), the smaller the utilities.

Given this transformation, we then design two polynomial-time algorithms for solving these infinite knapsack problems. The first algorithm is 1/2-approx. algorithm that has a running time of \( O(n^2(\log n + K)) \). The second algorithm is a fully polynomial time approximation scheme (FPTAS) with a running time of \( O(n^2K^3/\epsilon) \) for any \( \epsilon > 0 \). Note that the 1/2-approx. algorithm and FPTAS with the help of the Ellipsoid method lead to 1/2 and \( (1 - \epsilon) \) polynomial-time algorithms for the original Problem (FAIR). We refer to the resulting algorithms as the 1/2-approx. fair Ellipsoid-based and FPTAS fair Ellipsoid-based algorithms. Note that while theoretically our FPTAS attains a better approximation ratio than the 1/2-approx. algorithm, we show in numerical studies (Section 6) that the attained market share of these algorithms are comparable. In realistic scenarios, the 1/2-approx. algorithm also has a much better running time than the FPTAS, making it a viable practical solution.

1/2-approx. algorithm for an infinite series of parameterized knapsack problems. In Section 5.2, we present an 1/2-approx. algorithm (Algorithm 1) for an infinite series of parameterized knapsack problems. We first note that for each one of the aforementioned knapsack problems, a 1/2-approx. solution can be obtained using the profile associated with the optimal solution to the
LP relaxation of the knapsack problem, where the profile determines the set of items that are fully added, fractionally added, and not added to knapsack (Caprara et al. 2000). However, while solving one relaxed knapsack problem is doable in polynomial time, we cannot possibly solve an infinite number of them in order to find the corresponding approximate solutions. A key observation that we make here is that while the 1/2-approx. solutions can vary as the parameter of the knapsack increases, their associated profiles undergo at most $O(nK)$ number of changes. This then allows us to maintain a collection of sets with a polynomial-size that contains a 1/2-approx. solution to the infinite series of parameterized knapsack problems. To maintain this collection, as one of the most important steps of our algorithm, we determine the start of the change point in the profiles dynamically, and once the change point is detected, we update the profile using a simple rule that takes advantage of the structure of the parameterized knapsack problems.

**FPTAS for an infinite series of parameterized knapsack problems.** In Section 5.3, we further present an FPTAS (Algorithm 2) for the aforementioned knapsack problems, based on a dynamic programming (DP) scheme. Again, while it is easy to design a DP-based FPTAS for a single knapsack problem with cardinality constraint (see Caprara et al. (2000)), it is not clear how to design a DP-based FPTAS for an infinite series of parameterized knapsack problems. Consider designing an FPTAS for an infinite series of parameterized knapsack problems when the parameter $W$ falls into an interval $I = [W_{\text{min}}, W_{\text{max}}]$. (Recall that $W$, which determines the capacity of the knapsack, also influences the utility of items in the knapsack problem.) Our idea is to use a DP-based scheme to solve the knapsack problem near-optimally for the smallest parameter $W$ in that interval, i.e., $W_{\text{min}}$. To design this DP-based scheme, we crucially (i) re-scale the utility of items by a factor proportional to the maximum utility of items at $W_{\text{min}}$, and (ii) collect all the returned sets that are feasible for the knapsack problem with the largest capacity, i.e., $W_{\text{max}}$.

**Numerical studies using real-world and synthetic data.** In Section 6, we numerically evaluate our 1/2-approx. and FPTAS fair Ellipsoid-based algorithms on both real-world and synthetic datasets. In Section 6.1, we consider a movie recommendation setting based on the MovieLens dataset and investigate the performance of our near-optimal algorithms under different levels of fairness constraints. In Section 6.2, we reproduce the experiments in a number of synthetically generated, yet real-world inspired settings. In our numerical studies, we observe that while the market share of these two algorithms is quite comparable with one another, in realistic settings, the 1/2-approx. fair Ellipsoid-based algorithm has significantly better running time than the FPTAS fair Ellipsoid-based algorithm. Our studies further shed light on the price of fairness (i.e., the loss in market share due to enforcing fairness constraints), and the impact of fairness constraints on sellers’ visibility.

Proofs of all analytical results presented in this paper are included in the appendix.
2. Related Work

**Fairness in supervised learning.** With the increasing penetration of machine learning algorithms, there has been an explosion of work on algorithmic fairness in supervised learning, especially binary classification and risk assessment (Calders et al. (2009), Dwork et al. (2012), Hardt et al. (2016), Goel et al. (2018), Ustun et al. (2019)). Generally, the definitions of algorithmic fairness fall under three different types: (i) individual fairness, where similar individuals get similar predictions (Dwork et al. (2012), Kusner et al. (2017)), (ii) group fairness, where different groups are treated equally (Calders et al. (2009), Zliobaite (2015), Hardt et al. (2016)), and (iii) subgroup fairness, which combines individual and group notions of fairness (Kearns et al. (2018), Kearns et al. (2019)). Roughly speaking, in our setting, we consider individual fairness for each of the seller on the online platform. However, our work is different from the aforementioned works because we do not study a prediction problem using labeled data (as they do), but instead we look into an optimization problem (assortment optimization) while imposing fairness constraints.

**Fairness in resource allocation.** At a high level, in our problem, an online platform would like to allocate exposure (resource) to different items in a fair fashion while optimizing for customers’ satisfaction. Having this view in mind, there are works that study fair resource allocation problems for more tangible resources. These works can be divided into two streams. The first stream considers static settings where the resource allocation needs to be done in a single shot, and studies the trade-off between fairness and efficiency (Bertsimas et al. (2011), Bertsimas et al. (2012), Hooker and Williams (2012), Donahue and Kleinberg (2020)). (See also Cohen et al. (2021) for a recent work that studies fairness in price discrimination in a static setting.) The second stream investigates dynamic settings, where resources should be allocated over time. See Manshadi et al. (2021) for fairness in an online scarce resource allocation problem with correlated demands, Balseiro et al. (2021) and Bateni et al. (2021) for regularized fair online resource allocation problems, Ma et al. (2021) for fairness in online matching, Mulvany and Randhawa (2021) for fairness in scheduling, and Baek and Farias (2021) for fairness in bandit setting. Our work is closer to the works in the first stream, rather the second one. That being said, we are the first one that studies fairness in an assortment planning problem.

**Fairness in ranking.** Our work is also related to the literature on fairness in ranking, where the goal in a ranking problem is to optimize over the permutations of items displayed to users. We can view our assortment planning problem as a special case of ranking problem where the items placed in the top $K$ positions get visibility one (chosen in the assortment) and the rest get zero visibility (not chosen). Within this literature, Singh and Joachims (2018), Biega et al. (2018), Singh and Joachims (2019) are three works that are the most similar to ours in the sense that they oppose the winner-take-all allocation of economic opportunity. Like our work, all of these works also want
each item’s exposure/visibility to be proportional to its relevance/quality. Since the nature of both
ranking and assortment planning problems is combinatorial, designing an efficient algorithm is
challenging. Singh and Joachims (2018) overcome this by considering distribution over rankings and
imposing that the expected utility is linear in terms of both (i) attention an item gets at a certain
position, and (ii) utility value of each item for each type of user (position independent). Biega et al.
(2018) look at the dynamic settings over several rounds, where on each round, they utilize integer
LP to find the ranking in each round that minimizes cumulative measure of unfairness, given all of
the previous rankings. On the other hand, Singh and Joachims (2019) try to learn a fair ranking
using a policy-gradient approach that maximizes utility while minimizing group fairness disparity.
Both Biega et al. (2018) and Singh and Joachims (2019) do not have theoretical guarantees. We,
on the other hand, present a 1/2-approx. algorithm and a FPTAS with theoretical guarantees.

3. Model

We consider a platform with \( n \) items, indexed by \( i \in [n] \). Each item \( i \) has a (rational) weight \( w_i \geq 0 \)
that measures how popular the item is to the platform’s users. Upon offering assortment/set \( S \)
of items, the users purchase/choose item \( i \in S \) according to the MNL model (Train (1986)), with
probability \( \frac{w_i}{1 + w(S)} \), where \( w(S) = \sum_{i \in S} w_i \). The users may also choose not to purchase any item
with probability \( \frac{1}{1 + w(S)} \); that is, the no-purchase option (item 0) is always included in set \( S \). Here,
the action of purchasing an item can be viewed as “taking the desired action” under different
contexts (e.g., purchasing a product on an online retail store, clicking on a post on social media,
or watching a movie on a movie recommendation site).

3.1. Platform’s Objective

We consider maximizing market share as the platform’s primary objective. Here, the platform’s
market share is the probability that users make a purchase, which is equal to \( \frac{w(S)}{1 + w(S)} \) under assortment \( S \). The platform would like to optimize over its offered assortment that can contain at most
\( K \) items in order to maximize its market share. For many online retail platforms, maximizing
their market share is their prime objective, as they seek to build a long-run reputation of helping
customers find their desired items. (See, for example, Wang and Sahin (2018), Derakhshan et al.
(2020), Golrezaei et al. (2021b), Niazadeh et al. (2021) for works that use similar objectives for
the platform.) For other platforms such as social media platforms or job recommendation sites,
maximizing market share (or users’ engagement) is also considered their main goal.

In a more general setting where the sale of each item comes with certain profit, the platform can
alternatively consider maximizing revenue as its objective. Under such a setting, the sale of item \( i \)
generates profit \( r_i > 0 \), and the expected revenue upon offering assortment \( S \) is \( \text{REV}(S) = \frac{\sum_{i \in S} r_i w_i}{1 + \sum_{i \in S} w_i} \).
The platform seeks to optimize over its offered assortment to maximize its expected revenue. In the
following sections, we first focus on studying the fair assortment planning problem with maximizing market share as the primary objective. Later in Section 7, we will comment on how our near-optimal algorithms can be extended to such a more general platform’s objective.

3.2. Fairness Notion

The platform is also interested in providing a fair marketplace for all items. In particular, the platform considers a parameterized notion of fairness that is based on parity of pairwise visibility across different items. Let \( p(S) \) be the probability that the platform offers set \( S \) with \( |S| \leq K \). (For any \( S \) with \( |S| > K \), \( p(S) = 0 \).) Then, the visibility of item \( i \), denoted by \( v_i \), is defined as \( \sum_{S:i \in S} p(S) \). We say the platform is \( \delta \)-fair if for any pair of items \((i, j)\), we have

\[
\frac{w_j}{w_i} v_i - v_j \leq \delta \quad \Rightarrow \quad \frac{w_j}{w_i} \sum_{S:i \in S} p(S) - \sum_{S:j \in S} p(S) \leq \delta \quad i, j \in [n].
\] (1)

Note that as \( \delta \) increases, the platform gets less concerned about being fair toward the items. When \( \delta = 0 \) (i.e., when the platform is 0-fair), the above constraint can be written as \( \frac{w_j}{w_i} v_i = \frac{w_j}{w_i} v_j \). That is, under these constraints, the visibility of item \( i \) is proportional to its weight. For any \( \delta \geq \frac{\max_{i \in [n]} w_i}{\min_{i \in [n]} w_i} \), the fairness constraint is simply satisfied at the optimal solution in the absence of the fairness constraint, i.e., at the solution that offers the top \( K \) items with the largest weights. Note that while the optimal solution in the absence of the fairness constraint maximizes platform’s market share, it can lead to a very unfair outcome when the weight of some of items that are not included in the offered assortment are close to the weight of those included.

As stated earlier, our pairwise notion of fairness bears some resemblance to the fairness notion used in Singh and Joachims (2018), Biega et al. (2018), Singh and Joachims (2019) for ranking problems, not the assortment planning problem we study here. In these works, similar to our setting, roughly speaking, each item’s visibility is required to be proportional to its relevance/quality.

Our pairwise notion of fairness is in fact very flexible and can impose fairness with respect to any proxy of quality/merits of items. For instance, consider the more general setting where each item has profit \( r_i \), and the platform measures the quality of item \( i \) using some function \( q_i(w_i, r_i) \) that depends on both the popularity weight and the profit. Then, we can similarly require items with similar quality to have similar visibility, by imposing the following fairness constraints instead:

\[
\frac{q_j(w_j, r_j)}{q_i(w_i, r_i)} \sum_{S:i \in S} p(S) - \sum_{S:j \in S} p(S) \leq \delta \quad i, j \in [n], i \neq j.
\] (2)

In Section 7, we will see that our Ellipsoid-based framework can be readily extended to this more generic notion of fairness regardless of the proxy of quality chosen.
3.3. Platform’s Problem

Having defined the platform’s objective and the fairness notion, we now present the platform’s optimization problem:

\[
\text{FAIR} = \max_{p(S) \geq 0, S \subseteq [n]} \sum_{S: |S| \leq K} p(S) \cdot MS(S)
\]

\[
\text{s.t. } \frac{w_j}{w_i} \sum_{S: i \in S} p(S) - \sum_{S: j \in S} p(S) \leq \delta \quad i, j \in [n], i \neq j
\]

\[
\sum_{S: |S| \leq K} p(S) \leq 1 \quad \text{(FAIR)}
\]

where \(MS(S) = \frac{w(S)}{1 + w(S)}\) is the platform’s market share under set \(S \subseteq [n]\). Here, we slightly abuse the notation and denote both the platform’s problem and the optimal market share with \(\text{FAIR}\). In Problem (\(\text{FAIR}\)), the objective is the expected platform’s market share. The first set of constraints is the fairness constraints, and the second constraint enforces \(\{p(S)\}_{S: |S| \leq K}\) to be a probability distribution. Our goal in this paper is to provide computationally-efficient algorithms with provable performance guarantees for Problem (\(\text{FAIR}\)) with any \(\delta \geq 0\). Observe that this problem is always feasible for any \(\delta \geq 0\). In particular, the following solution is always feasible: \(p(\{i\}) = \frac{w_i}{w([n])}\) and \(p(S) = 0\) for any \(S\) with \(|S| > 1\) and \(S = \emptyset\).

4. Near-optimal Algorithm for Problem (FAIR)

In this section, we start by presenting some main properties of Problem (\(\text{FAIR}\)) that motivate the design of our algorithms. We then present a general framework that allows us to provide a wide-range of near-optimal algorithms for Problem (\(\text{FAIR}\)), which relies on the Ellipsoid method and the dual counterpart of Problem (\(\text{FAIR}\)).

4.1. Main Properties of Problem (FAIR)

As we discussed earlier, without the fairness constraints, Problem (\(\text{FAIR}\)) admits a simple optimal solution under which a single set is offered to users. Let \(S_K\) be the set containing the top \(K\) items with the highest weight. Then, without the fairness constraints, the optimal solution to Problem (\(\text{FAIR}\)) sets \(p(S_K) = 1\) and \(p(S) = 0\) for any other sets \(S \neq S_K\). However, such a solution may not be feasible to Problem (\(\text{FAIR}\)) in the presence of fairness constraints. To see that, consider item \(i \in S_K\) and \(j \notin S_K\). Then, in the aforementioned solution, while item \(i\) gets full visibility (i.e., \(v_i = 1\)), item \(j\) gets zero visibility (i.e., \(v_j = 0\)), and hence Constraint (1) (i.e., \(\frac{w_j}{w_i}v_i - v_j \leq \delta\)) is not satisfied for a sufficiently small \(\delta\).

This discussion sheds light on the fact that an optimal solution to Problem (\(\text{FAIR}\)) may involve randomization over multiple sets, which might lead to difficulties when implementing this in practice. However, those concerns can be alleviated, as we show in the following theorem that for any
\( \delta \geq 0 \), there exists an optimal solution to Problem (FAIR) that involves randomization over at most \( O(n^2) \) sets, easing implementation of the optimal solution.

**Theorem 1 (Randomization over at most \( O(n^2) \) sets).** For any \( \delta \geq 0 \), there exists an optimal solution \( p^*(\cdot) \) to Problem (FAIR) such that
\[
|\{S : p^*(S) > 0\}| \leq n(n - 1) + 1.
\]

Motivated by Theorem 1 above, we aim to design near-optimal algorithms for Problem (FAIR) that also randomize over a polynomial number of sets. We proceed to discuss our framework for finding such near-optimal solutions in Sections 4.2 and 4.3.

### 4.2. Dual Counterpart of Problem (FAIR)

Our framework relies on the dual counterpart of Problem (FAIR), which can be written as

\[
\text{FAIR-dual} = \min_{\rho \geq 0, z \geq 0} \rho + \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z_{ij} \right)
\]

\[
\text{s.t. } \sum_{i \in S} \left( \sum_{j=1, j \neq i}^{n} z_{ij} w_j - \sum_{j=1, j \neq i}^{n} z_{ji} w_j \right) + \rho \geq \text{MS}(S), \quad \forall S : |S| \leq K.
\]

(FAIR-dual)

Here, \( z_{ij} \)'s (\( i, j \in [n], i \neq j \)) are the dual variable of the first set of constraints and \( \rho \geq 0 \) is the dual variable of the second constraint in Problem (FAIR). Note that \( z_{ij} \) can be viewed as the 

\textit{fairness cost} that item \( i \) has caused due to the presence of item \( j \) on the platform. We call each constraint in Problem (FAIR-dual) \textit{a dual fairness constraint} on set \( S \). Observe that the set of dual fairness constraints in Problem (FAIR-dual) can be written as the following single constraint:

\[
\rho \geq \max_{S : |S| \leq K} \left\{ \text{MS}(S) - \sum_{i \in S} \left( \sum_{j=1, j \neq i}^{n} z_{ij} w_j - \sum_{j=1, j \neq i}^{n} z_{ji} w_j \right) \right\} = \max_{S : |S| \leq K} \text{MS-cost}(S, z).
\]

(3)

Here, we refer to \( \text{MS-cost}(S, z) \) as the \textit{cost-adjusted market share}, where

\[
\text{MS-cost}(S, z) = \text{MS}(S) - \sum_{i \in S} c_i(z),
\]

(4)

and for dual vector \( z \), for any \( i \in [n] \), we define \( c_i(z) \) as the \textit{cost} of item \( i \):

\[
c_i(z) = \sum_{j=1, j \neq i}^{n} z_{ij} w_j - \sum_{j=1, j \neq i}^{n} z_{ji} w_j
\]

(5)

Note that the cost of item \( i \) is a linear combination of the fairness costs \( z_{ij} \) and \( z_{ji} \), \( j \neq i \). More specifically, when item \( i \) incurs the fairness cost of \( z_{ij} \) for some \( j \neq i \), the cost of item \( i \) increases by \( z_{ij} w_j \). On the other hand, when item \( j \) causes the fairness cost of \( z_{ji} \), the cost of item \( i \) decreases by \( z_{ji} w_j \). Observe that in the optimal solution of Problem (FAIR-dual), the inequality in Equation (3) must be tight, i.e., it is satisfied with equality.

The following theorem shows that the optimization problem in Equation (3) is NP-complete.
Theorem 2 (NP-completeness). Consider the following problem that represents the first set of constraints in Problem (FAR-DUAL):

\[
\text{SUB-DUAL}(\mathbf{z}, K) = \max_{S:|S| \leq K} \text{MS-COST}(S, \mathbf{z}),
\]

where MS-COST(S, \mathbf{z}) is defined in Equation (4). Problem (\text{SUB-DUAL}(\mathbf{z}, K)) is NP-complete.

To show this, we consider an arbitrary instance of the partition problem, which is known to be NP-complete (Hayes 2002). We reduce this instance of the partition problem to a polynomial number of instances of Problem (\text{SUB-DUAL}(\mathbf{z}, K)), showing Problem (\text{SUB-DUAL}(\mathbf{z}, K)) is NP-complete.

4.3. Fair Ellipsoid-based Framework

We now present our framework for obtaining near-optimal solutions to Problem (FAIR), which relies on the Ellipsoid method and the dual problem (FAR-DUAL). We will show that to obtain a near-optimal (\(\beta\)-approximate) solution to Problem (FAIR) that randomizes over a polynomial number of assortments, it suffices to have a \(\beta\)-approximate solution to Problem (\text{SUB-DUAL}(\mathbf{z}, K)). See Theorem 3 stated at the end of this section. In particular, the approximation algorithm for Problem (FAIR-DUAL) will play the role of an approximate separation oracle for the Ellipsoid method that we apply to the dual problem (FAIR-DUAL). Assume that we have access to a polynomial-time algorithm \(A\) that gives a \(\beta\)-approximate solution for (\text{SUB-DUAL}(\mathbf{z}, K)), for any \(\mathbf{z} \geq 0\) and \(K \in [n]\).

We propose a fair Ellipsoid-based algorithm that consists of the following two steps:

**Step 1:** Apply the Ellipsoid method to the dual problem (FAIR-DUAL) using a \(\beta\)-approximate separation oracle. At a high level, the Ellipsoid method generates a sequence of ellipsoids whose volume decreases by a constant factor in each iteration. In each iteration, the method determines whether the center of the current ellipsoid (i.e., the current solution to the dual problem) is within the feasibility region of the dual problem (FAIR-DUAL). If the current solution is feasible, the method then attempts to find a smaller ellipsoid whose center further decreases the objective function. If it is not feasible, a violated constraint is returned by a separation oracle, and we proceed to examine a smaller ellipsoid whose center satisfies this particular constraint. The Ellipsoid method stops when the ellipsoid is sufficiently small, which means that we can no longer find a feasible solution with a smaller objective. That is, we have found an optimal feasible solution.

In each iteration of the Ellipsoid method, a polynomial-time approximate separation oracle, which makes use of the approximation algorithm \(A\) for (\text{SUB-DUAL}(\mathbf{z}, K)), checks whether the current solution \((\mathbf{z}, \rho)\) is feasible, and, if not, finds a violated constraint. (Note that Problem (FAIR-DUAL) has exponentially many constraints and hence simply examining each of the constraints in (FAIR-DUAL) will result in an exponential-time algorithm.) Recall that the first set of constraints in Problem (FAIR-DUAL) can be written as \(\rho \geq \max_{S:|S| \leq K} \text{MS-COST}(S, \mathbf{z}) =\)
\textsc{sub-dual}(z, K), and as we show in Theorem 2, the Problem \textsc{sub-dual}(z, K) is \NP-complete. As a result, to examine the dual fairness constraints in Problem (\textsc{fair-dual}), we instead apply the following approximate separation oracle to the current dual solution (z, ρ):

1. We first apply the polynomial-time β-approximate algorithm \( A \) to Problem (\textsc{sub-dual}(z, K)), which returns \( S_A \) with \(|S_A| \leq K\) such that \( \text{ms-cost}(S_A, z) \geq \beta \cdot \text{sub-dual}(z, K) \).²

2. We then determine whether the current solution (z, ρ) is feasible. If \( \text{ms-cost}(S_A, z) > \rho \), the separation oracle has found the violating constraint: set \( S_A \) violates the dual fairness constraint. If \( \text{ms-cost}(S_A, z) \leq \rho \), the separation oracle will declare the solution (z, ρ) feasible.

In Appendix B, we present the Ellipsoid method for Problem (\textsc{fair-dual}) used in Step 1; see Algorithm 3 for more details.

**Step 2:** Solve Problem (\textsc{fair}) using only sets for which the dual fairness constraint is violated.

Throughout the run of the Ellipsoid method, we keep track of all the sets that have violated the dual fairness constraint at some iteration. Let \( V \subseteq \{S : |S| \leq K\} \) be the collection of sets for which the dual fairness constraint has been violated in the Ellipsoid method. We will show, in the proof of Theorem 3, that \(|V|\) is polynomial in \( n \) and \( w_{\text{max}} \). Here, \( w_{\text{max}} \in \mathbb{Z}^+ \) is the maximum adjusted weights, where, for any rational weight \( w_i \), its adjusted version is obtained by multiplying it by the smallest common denominator of all the weights. We then solve the primal problem (\textsc{fair}) by additionally setting \( p(S) = 0 \) for all \( S \notin V \). This reduces the number of non-zero variables in Problem (\textsc{fair}) to a polynomial size, and allows us to solve it in polynomial time.

In the following theorem, we show that the fair Ellipsoid-based algorithm above (i) gives a \( \beta \)-approximate solution for Problem (\textsc{fair}), which only randomizes over a polynomial number of assortments, and (ii) runs in polynomial time.

**Theorem 3 (\( \beta \)-approximation algorithm for Problem (\textsc{fair})).** Suppose that for any \( z \geq 0 \) and \( K \in [n] \), we have a polynomial \( \beta \)-approximate algorithm \( A \) for Problem (\textsc{sub-dual}(z, K)). That is, \( A \) returns set \( S_A \) with \(|S_A| \leq K\) such that

\[
\text{ms-cost}(S_A, z) \geq \beta \cdot \text{sub-dual}(z, K)
\]

Then, there exists a polynomial algorithm that returns a \( \beta \)-approximate feasible solution \( \hat{p}(S) \), \( S \subseteq [n] \) to Problem (\textsc{fair}). That is, \( \sum_{S : |S| \leq K} \hat{p}(S) \cdot \text{ms}(S) \geq \beta \cdot \text{fair} \) and the returned solution is \( \delta \)-fair: \( \frac{w_i}{w_i} \sum_{S : i \in S} \hat{p}(S) - \sum_{S : j \in S} \hat{p}(S) \leq \delta \) for any \( i, j \in [n] \). In addition, the number of sets \( S \) such that \( \hat{p}(S) > 0 \) is polynomial in \( n \) and \( \log(w_{\text{max}}) \), where \( w_{\text{max}} \in \mathbb{Z}^+ \) is the maximum adjusted weight.

²Before checking the dual fairness constraints, we would first check the non-negativity constraints, to ensure that \( \rho \geq 0 \) and \( z \geq 0 \) when the algorithm \( A \) is invoked. See details in Appendix B.
In light of Theorems 2 and 3, our goal in the next section is to present a polynomial-time $\beta$-approximate algorithm for Problem $(\text{SUB-DUAL}(z, K))$. We note that the objective function of Problem $(\text{SUB-DUAL}(z, K))$ is a non-monotone submodular function, and hence, the continuous double greedy algorithm proposed in Buchbinder et al. (2014) will give a $[1/e + 0.004, 1/2]$-approx. solution to Problem $(\text{SUB-DUAL}(z, K))$.\footnote{In Buchbinder et al. (2014), the continuous double algorithm is shown to achieve an approximation ratio of $1/2 - o(1)$ when $K = n/2$; for $K < n/2$, the algorithm is guaranteed to have an approximation ratio of $1/e + 0.004$.} However, as we will show in the next section, given the special structure of our problem, we can do much better than an $1/e$-approximation for any value of $K$. In particular, we will present a $1/2$-approx. algorithm, as well as, an FPTAS for Problem $(\text{SUB-DUAL}(z, K))$, which obtains a $(1 - \epsilon)$-approximate solution for any $\epsilon > 0$.

5. Near-optimal Algorithms for Problem $(\text{SUB-DUAL}(z, K))$

In this section, we propose two near-optimal algorithms for Problem $(\text{SUB-DUAL}(z, K))$. In Section 5.2, we first present a $1/2$-approx. algorithm for Problem $(\text{SUB-DUAL}(z, K))$. By plugging this algorithm in the fair Ellipsoid-based algorithm presented in Section 4.3, we have a $1/2$-approx. algorithm for Problem $(\text{FAIR})$; see Theorem 3. We refer to this algorithm as $1/2$-approx. fair Ellipsoid-based algorithm. Then, in Section 5.3, we present an FPTAS. Again, by plugging this algorithm in the fair Ellipsoid-based algorithm, we have an FPTAS for problem $(\text{FAIR})$. We refer to this algorithm as FPTAS fair Ellipsoid-based algorithm. The design of both the $1/2$-approx. algorithm and the FPTAS crucially uses a transformation that shows Problem $(\text{SUB-DUAL}(z, K))$ is equivalent to an infinite series of parameterized knapsack problems.\footnote{We remark that a similar transformation is adopted in Kunnumkal et al. (2010). However, the problem that the authors study is different from ours. Further, in their setting the knapsack problem does not come with any cardinality constraints, which is the most challenging aspect in our problem.} We provide more details about this transformation in Section 5.1.

From a theoretical point of view, our FPTAS fair Ellipsoid-based algorithm attains a better approximation factor than the $1/2$-approx. fair Ellipsoid-based algorithm. However, we will show, in our numerical studies (Section 6), that the empirical performance of the two algorithms are in fact comparable in terms of both the market share and the number of sets they randomize over. The $1/2$-approx. algorithm also demonstrates its value in practical applications with its fast computational performance.

\footnote{A set function $f : 2^{[n]} \to \mathbb{R}$ is submodular if for any $A, B \subseteq [n]$, we have $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$. It is monotone if for any $A \subseteq B \subseteq [n]$, we have $f(A) \leq f(B)$.}
5.1. Transforming Problem (SUB-DUAL(z, K)) to an Infinite Series of Knapsack Problems

Here, we transform Problem (SUB-DUAL(z, K)) into an infinite series of parameterized knapsack problems with cardinality constraints. Consider the following knapsack problem with capacity $W \geq 0$ and cardinality constraint $K$:

\[
\text{KP}(W, K) = \max_S \frac{\sum_{i \in S} (w_i - c_i)}{1 + W}
\]

\[
\text{s.t. } \sum_{i \in S} w_i \leq W
\]

\[
|S| \leq K
\]

Here, the **utility** of item $i \in [n]$ is $\frac{w_i}{1+W} - c_i$ and its weight is $w_i$. Further, recall that the cost of item $i$ is $c_i = \sum_{j=1, j \neq i}^{n} z_{ij} w_j - \sum_{j=1, j \neq i}^{n} z_{ji} w_j$. Since throughout Section 5, the dual vector $z$ is fixed, we suppress the dependency of the variables on $z$. For a given $W$, we denote the utility of item $i$ (i.e., $\frac{w_i}{1+W} - c_i$) with $u_i(W)$. Observe that in $(\text{KP}(W, K))$, the first constraint is the capacity constraint and the second constraint is the cardinality constraint. The former ensures that the sum of the weight of the items that are added to the knapsack does not exceed $W$ and the latter ensures that at most $K$ items are added to the knapsack.

The following theorem characterizes the relationship between Problems $(\text{KP}(W, K))$, $W \geq 0$, and Problem (SUB-DUAL(z, K)).

**Theorem 4** (Problem (SUB-DUAL(z, K)) as an infinite series of knapsack problems).

*For any $z \geq 0$ and $K \in [n]$, we have

\[
\text{SUB-DUAL}(z, K) = \max_{W \geq 0} \text{KP}(W, K).
\]

In light of Theorem 4, to design approximation algorithms for Problem (SUB-DUAL(z, K)), we can instead design:

1. a $1/2$-approx. algorithm for problem $\max_{W \geq 0} \text{KP}(W, K)$ that finds a set $S$ such that $\text{MS-COST}(S, z) \geq \frac{1}{2} \max_{W \geq 0} \text{KP}(W, K)$, and runs in time polynomial in the input size.

2. An FPTAS for problem $\max_{W \geq 0} \text{KP}(W, K)$ that finds a set $S$ such that $\text{MS-COST}(S, z) \geq (1 - \epsilon) \max_{W \geq 0} \text{KP}(W, K)$, for any $\epsilon > 0$, in time polynomial in both the input size and $1/\epsilon$.

Observe that the problem $\max_{W \geq 0} \text{KP}(W, K)$ requires us to solve an infinite number of cardinality-constrained knapsack problems, i.e., $\text{KP}(W, K)$ for any $W \geq 0$. Further note that in the cardinality-constrained knapsack problem $\text{KP}(W, K)$, as we increase $W$, the capacity of the knapsack enlarges, but the utility of the items decreases. (Recall that the utility of item $i$ is $\frac{w_i}{1+W} - c_i$). Hence, it is unclear which value of $W$ maximizes $\text{KP}(W, K)$ and more importantly, if our transformation has simplified the problem. As we will show in Sections 5.2 and 5.3, this transformation in conjunction with taking advantage of the structure of cardinality-constrained knapsack problems $\text{KP}(W, K)$ as a function of $W$ leads to near-optimal algorithms for the original Problem (SUB-DUAL(z, K)).
5.2. 1/2-Approx. Algorithm for Problem (SUB-DUAL(z, K))

In this subsection, we first propose a 1/2-approx. algorithm for Problem (SUB-DUAL(z, K)), by using the transformation in Theorem 4. In the transformed problem, we need to optimize over the capacity of knapsack $W$, where $W$ can take any value in the interval $[0, \infty)$. At a high level, our 1/2-approx. algorithm maintains a collection $C$ of assortments such that for any $W \in [0, \infty)$, the collection $C$ contains an 1/2-approx. feasible solution to Problem $(\text{KP}(W,K))$. To design a 1/2-approx. algorithm to this series of infinite knapsack problems, we take advantage of the LP relaxation of the original knapsack problem $(\text{KP}(W,K))$, defined below

$$\text{KP-relax}(W,K) = \max_{x \in [0,1]^n} \sum_{i \in [n]} \left( \frac{w_i}{1 + W} - c_i \right) x_i$$

s.t. $\sum_{i \in [n]} w_i x_i \leq W$ and $\sum_{i \in [n]} x_i \leq K$. (KP-relax$(W,K)$)

Observe that if we enforce $x_i \in \{0,1\}$, rather than $x_i \in [0,1]$, we recover Problem $(\text{KP}(W,K))$. The variable $x_i$, $i \in [n]$, can be viewed as the fraction of item $i$ that we fit in the knapsack. When we talk about integer solutions to this problem, we sometimes use sets instead of $x \in [0,1]^n$. For an integer solution $x \in \{0,1\}^n$, the corresponding set is $\{i \in [n] : x_i = 1\}$.

For any fixed $W \in [0, \infty)$, the relaxed problem KP-relax$(W,K)$ always admits an optimal basic (feasible) solution.\(^6\) In Lemma 1 in Caprara et al. (2000), it is shown that any optimal basic solution to Problem KP-relax$(W,K)$ has at most two fractional variables.\(^7\) Let $x^*(W)$ be an optimal basic solution to Problem KP-relax$(W,K)$ that has at most two fractional variables. We then define $P_1(W) = \{i \in [n] : x^*_i(W) = 1\}$ as the set of items that have a value of 1 in $x^*(W)$, which are the items fully added to the knapsack associated with Problem KP-relax$(W,K)$.\(^8\) Similarly, we define $P_f(W) = \{i \in [n] : 0 < x^*_i(W) < 1\}$ (respectively $P_0(W) = \{i \in [n] : x^*_i(W) = 0\}$) as the set of items that have a fractional value (respectively value of zero) in $x^*(W)$. (Here, “f” in $P_f$ stands for fractional.) As stated earlier, $|P_f(W)| \leq 2$ and hence, here we denote $P_f(W)$ by an ordered pair of items $(i,j)$. When the fractional set $P_f(W)$ is empty, both $i,j$ would be a dummy item and in particular, the no-purchase item 0. When the fractional set $P_f(W)$ has a single item, we set

\(^6\)See Appendix J for definition of a basic (feasible) solution. Here in Problem KP-relax$(W,K)$, since its feasibility region is nonempty ($x = 0$ is feasible) and bounded (since $x \in [0,1]^n$), by Corollary 2.2 in Bertsimas and Tsitsiklis (1997), it has at least one basic feasible solution. Further, because its optimal objective is also bounded, by Theorem 2.8 in Bertsimas and Tsitsiklis (1997), it admits at least one optimal basic feasible solution.

\(^7\)Note that in the absence of the cardinality constraint, the relaxed problem admits an optimal basic solution that has at most one fractional variable. In addition, in the optimal solution to the relaxed problem, we greedily fill up the knapsack with the items with the highest utility-to-weight ratio. However, this structure may not be optimal in the presence of the cardinality constraint.

\(^8\)Note that the optimal basic solution to Problem KP-relax$(W,K)$ (i.e., $x^*(W)$) and consequently set $P_1(W) = \{i \in [n] : x^*_i(W) = 1\}$ may not be unique. Given that one could present $P_1(W)$ with $P_1(x^*(W))$ instead. However, we avoid using this heavy notation in the paper to simplify the exposition.
we need to find approximate solutions for an infinite number of problems. The key observation that we make here is that to find a 1/2-approx. solution to Problem \((\text{KP}(W,K))\), we can construct an integer 1/2-approx. solution to Problem \((\text{KP}(W,K))\), which also serves as an 1/2-approx. solution to Problem \((\text{KP}(W,K))\).

**Lemma 1 (1/2-Approx. Solutions to Problem (\text{KP-relax}(W,K)) (Caprara et al. 2000)).**

For a fixed \(W \in [0, \infty)\), let \(x^*(W)\) be an optimal basic solution to Problem \((\text{KP-relax}(W,K))\) and let \(P(W) = \{P_1(W), P_f(W), P_0(W)\}\) be its associated profile, where \(P_f(W) = (i,j)\). We have one of the following:

1. **No fractional variable.** If \(P_f(W) = (0,0)\), \(P_1(W)\) is an integer optimal solution to Problem \((\text{KP-relax}(W,K))\).
2. **One fractional variable.** If \(P_f(W) = (i,0)\) for some \(i \in [n]\), either \(P_1(W)\) or \(\{i\}\) is an integer 1/2-approx. solution to Problem \((\text{KP-relax}(W,K))\).
3. **Two fractional variables.** If \(P_f(W) = (i,j)\) for some \(i,j \in [n]\) such that \(w_i \leq w_j\), either \(P_1(W) \cup \{i\}\) or \(\{j\}\) is an integer 1/2-approx. solution to Problem \((\text{KP-relax}(W,K))\).

It is clear from Lemma 1 that for any given \(W \in [0, \infty)\), as long as we know the profile of the optimal basic solution to the relaxed problem \(\text{KP-relax}(W,K)\), we would be able to find a 1/2-approx. solution for Problem \((\text{KP}(W,K))\). However, in our problem, i.e., \(\max_{W \in [0, \infty)} \text{KP}(W,K)\), we need to find approximate solutions for an infinite number of knapsack problems, defined on the interval \(W \in [0, \infty)\). While solving one relaxed knapsack problem is doable in polynomial time, we cannot possibly solve an infinite number of them in order to find the corresponding approximate solutions. The key observation that we make here is that to find a 1/2-approx. solution of Problem \((\text{KP}(W,K))\) or \((\text{KP-relax}(W,K))\), the only information that we need is the profile \(P(W)\), instead of the actual optimal basic solution \(x^*(W)\) to Problem \((\text{KP-relax}(W,K))\). While \(x^*(W)\) might keep changing for different knapsack capacities \(W\), as we will show later in the proof of Theorem 5, their associated profile \(P(W)\) undergoes a polynomial number of changes as \(W\) changes. This allows us to consider a polynomial number of sub-intervals where in each sub-interval, the profile \(P(W)\) does not change, and we can consider the same collection of sets to construct a 1/2-approx. solution. Algorithm 1 is motivated by the idea outlined above, and returns an integer 1/2-approx. solution for \(\max_{W \in [0, \infty)} \text{KP-relax}(W,K)\), which also serves as an 1/2-approx. solution to Problem \(\max_{W \in [0, \infty)} \text{KP}(W,K)\). In particular, the algorithm maintains a collection of sets, denoted by \(C\) that contains all the sets that could be an integer 1/2-approx. solution to the problem \(\max_{W \in [0, \infty)} \text{KP-relax}(W,K)\).
For \( j = 1, \ldots, K \), we define \( H_j = \{ h_1, \ldots, h_j \} \subset [n] \) as the \( j \) items in \([n]\) with the highest utility-to-weight ratios, where the utility-to-weight ratio of item \( i \in [n] \) is
\[
\frac{u_i(W)}{w_i} = \frac{\frac{w_i}{1+w_i} - c_i}{w_i} = \frac{1}{1+W} - \frac{c_i}{w_i}.
\]
Here item \( h_i \) is the item with the \( i \)-th highest utility-to-weight ratio. Observe that the order of utility-to-weight ratio of items does not depend on the value of \( W \) and only depends on their costs \( c = (c_1, \ldots, c_n) \) and weights \( w = (w_1, \ldots, w_n) \). In Algorithm 1, we first partition \([0, \infty)\) into two intervals \( I_{\text{low}} = [0, W_{\text{th}}) \) and \( I_{\text{high}} = [W_{\text{th}}, \infty) \), where \( W_{\text{th}} = w(H_K) \) is the sum of the weights of the \( K \) items with the highest utility-to-weight ratios. We now discuss each of these intervals separately.

**Algorithm 1** 1/2-approx. algorithm for \( \max_{W \in [0, \infty)} \text{KP}(W, K) \)

**Input:** weights \( w = (w_1, \ldots, w_n) \), costs \( c = (c_1, \ldots, c_n) \), and cardinality upper bound \( K \).

**Output:** assortment \( S_{1/2} \).

1. **Initialization.**
   
   (a) Initialize the collection of assortments \( \mathcal{C} = \emptyset \). For \( j \in [n] \), add \( \{ j \} \) to \( \mathcal{C} \).
   
   (b) Rank the items by their utility-to-weight ratios. Let \( h_j \) be the index of the item with the \( j \)-th highest utility-to-weight ratio, for \( j \in [n] \), and define \( H_j = \{ h_1, \ldots, h_j \} \), for \( j \in [K] \) and \( W_{\text{th}} = w(H_K) \).

2. **Interval** \( I_{\text{low}} = [0, W_{\text{th}}) \).
   
   (a) For \( j = 1, \ldots, K \), add \( H_j \) to \( \mathcal{C} \).
   
   (b) **Stopping rule.** If \( u_{h_K}(W_{\text{th}}) \geq 0 \), go to Step 3; otherwise, go to Step 4 (i.e., the termination step).

3. **Interval** \( I_{\text{high}} = [W_{\text{th}}, \infty) \).
   
   (a) **Initialize the profile.** Set \( P_1 = H_K, P_0 = [n] \setminus H_K \), and \( W_{\text{next}} = W_{\text{th}} \).
   
   (b) **Adaptively partitioning** \( I_{\text{high}} \). While there exist \( i \in P_1, j \in P_0 \) such that \( w_i < w_j \):
      
      i. Update \( W_{\text{next}} \) and the indices \( i^*, j^* \) as follows:
      
      \[
      (i^*, j^*) \leftarrow \min_{i \in P_1, j \in P_0 \mid w_i < w_j} \frac{c_j - c_i}{w_j - w_i}, \quad \text{and} \quad W_{\text{next}} \leftarrow W_{\text{next}} - w_{i^*} + w_{j^*}.
      \]
      
      ii. **Swapping the two items.** Set \( P_1 \leftarrow P_1 \cup \{ j^* \} \setminus \{ i^* \} \) and \( P_0 \leftarrow P_0 \cup \{ i^* \} \setminus \{ j^* \} \).
      
      iii. **Stopping rule.** If \( \frac{c_{j^*} - c_{i^*}}{w_{j^*} - w_{i^*}} \leq \frac{1}{1+W_{\text{next}}} \), add \( P_1 \) to \( \mathcal{C} \). Otherwise, go to the termination Step 4.

4. **Termination Step.** Return \( S_{1/2} = \arg \max_{S \in \mathcal{C}} \text{MS-COST}(S, z) \).

Interval \( I_{\text{low}} = [0, W_{\text{th}}) \). For any \( W \in I_{\text{low}} = [0, W_{\text{th}}) \), solving the relaxed problem \( \text{KP-RELAX}(W, K) \) is rather straightforward because the solution is simply filling up the knapsack
with the items with the highest utility-to-weight ratios until we reach the capacity \( W \), regardless of the value of \( W \). This is the case, because for any \( W \in I_{low} \), the cardinality constraint is not binding, and hence in an optimal basic solution to the relaxed problem, we can greedily fill out the knapsack with the items that have the highest utility-to-weight ratios. (Recall that \( W_{th} = w(H_K) \).) Therefore, the profile of the optimal basic solution to the relaxed problem, i.e., \( \mathcal{P}(W) = \{ P_1(W), P_f(W), P_0(W) \} \), always has the following structure:

\[
P_1(W) = H_{j-1} \text{ and } P_f(W) = (h_j, 0) \text{ for some } j \in [K].
\]

That explains why in Step 2 of the algorithm, we add sets \( H_j, j = 1, \ldots, K \), to the collection \( \mathcal{C} \).

**Interval** \( I_{high} = [W_{th}, \infty) \). Before discussing how this interval can be handled, we make two remarks. First, in some cases, interval \( I_{high} = [W_{th}, \infty) \) does not need to get examined at all. In particular, when \( u_{h_K}(W_{th}) \leq 0 \), where \( h_K \) is the item with the \( K \)-th highest utility-to-weight ratio, it is easy to see that at \( W_{th} \), there are less than \( K \) items with positive utilities. Therefore, \( \arg \max_{W \in [0, \infty)} \text{KP-RELAX}(W, K) \) must be less than \( W_{th} \) and hence interval \( I_{high} \) is no longer relevant; see Step 2b of Algorithm 1. Second, while interval \( I_{high} \) ranges from \( W_{th} \) to infinity, Algorithm 1 does not investigate all \( W \)’s in this range; see Step 3(b)iii. In this step, the algorithm checks a simple inequality: when this inequality fails to hold, the algorithm knows, increasing \( W \) any further would not be helpful as for such \( W \)’s, the capacity constraint at the relaxed problem \( \text{KP-RELAX}(W, K) \) is no longer binding, and hence, as shown in Lemma 3, such \( W \) cannot be a maximizer of the optimization problem \( \max_{W \in [0, \infty)} \text{KP-RELAX}(W, K) \).

We are now ready to discuss interval \( I_{high} \). For \( W \in I_{high} = [W_{th}, \infty) \), solving the relaxed problem \( \text{KP-RELAX}(W, K) \) becomes more difficult because the profile of the optimal basic solution to the relaxed problem no longer has the nice, pre-specified structure that it had on interval \( I_{low} \). Intuitively, for large capacity \( W \), the cardinality constraint gets binding. With a binding cardinality constraint, it is no longer optimal to fill out the knapsack problem associated with the relaxed problem \( \text{KP-RELAX}(W, K) \) with items that have the highest utility-to-weight ratios. To handle interval \( I_{high} \), we partition \( I_{high} \) into a polynomial number of sub-intervals, such that on each sub-interval, the profile \( \mathcal{P}(W) \) does not change, and hence by Lemma 1, we can consider the same approximate solution(s) for this sub-interval. However, the main difficulty here is that these sub-intervals are not known in advance. To overcome this difficulty, in Algorithm 1, we determine these sub-intervals in an adaptive manner.

**Determine sub-intervals adaptively.** To do so, the algorithm keeps updating two quantities as it increases the capacity \( W \): (i) capacity *change points* denoted by \( W_{next} \), and (ii) two sets that represent the profile at \( W_{next} \). The first set, denoted by \( P_1 \), is \( P_1(W_{next}) \) that contains all the items
that are added completely to the knapsack in the relaxed knapsack problem \( \text{KP-RELAX}(W_{\text{next}}, K) \), and the second set, denoted by \( P_0 \), is \( P_0(W_{\text{next}}) \) that contains all the items that are not added to the knapsack in \( \text{KP-RELAX}(W_{\text{next}}, K) \). Note that \( W_{\text{next}} \) is chosen such that \( P_f(W_{\text{next}}) \) is empty; that is, \( P_f(W_{\text{next}}) = (0, 0) \). To update the sets \( P_1 \) and \( P_0 \), as well as, \( W_{\text{next}} \), the algorithm finds two items \( i^* \in P_1 \) and \( j^* \in P_0 \) by solving a simple optimization problem stated in Step 3(b)i. The algorithm then swap these two items—that is, it removes \( i^* \) from \( P_1 \) and adds it to \( P_0 \); similarly, it removes \( j^* \) from \( P_0 \) and adds it to \( P_1 \). This is, in fact, one of the main novel aspects of the algorithm that makes obtaining a 1/2-approx. solution in polynomial time possible.

We now provide more details about the aforementioned swapping idea. Let \( W_{\text{next}} \) and \( \hat{W}_{\text{next}} \) be two consecutive capacity change points. Further, let \( i^* \in P_1(W_{\text{next}}) = P_1 \) and \( j^* \in P_0(W_{\text{next}}) = P_0 \) with \( w_{i^*} < w_{j^*} \), be the solution to optimization problem in Step 3(b)i. That is, \((i^*, j^*) \in \arg \min_{i \in P_1, j \in P_0} (c_j - c_i)/(w_j - w_i) \). At a high level, by solving this optimization problem, we find two items \( i^*, j^* \) such that swapping them yields the highest marginal increase in utility. Then, after swapping, \( \hat{W}_{\text{next}} = W_{\text{next}} - w_{j^*} + w_{i^*} \) becomes the new capacity change point. For any \( W \in (W_{\text{next}}, \hat{W}_{\text{next}}) \), both items \( i^* \) and \( j^* \) are fractional items in the optimal basic solution for the relaxed problem \( \text{KP-RELAX}(W, K) \); that is, \( P_f(W) = (i^*, j^*) \). As \( W \) increases from \( W_{\text{next}} \) to \( \hat{W}_{\text{next}} \), the fraction of \( i^* \) (i.e., \( x^*_i(W) \)) decreases while the fraction of \( j^* \) (i.e., \( x^*_j(W) \)) increases; See Figure 1. This trend continues until \( x^*_i(W) \) hits zero and \( x^*_j(W) \) hits one, which happens at \( W = \hat{W}_{\text{next}} = W_{\text{next}} - w_{i^*} + w_{j^*} \). (Note that for all \( W \in [W_{\text{next}}, \hat{W}_{\text{next}}] \), we have \( x^*_i(W) + x^*_j(W) = 1 \); see Lemma 6.) This then leads to \( P_1(\hat{W}_{\text{next}}) = P_1(W_{\text{next}}) \cup \{j^*\} \setminus \{i^*\} \) and \( P_0(\hat{W}_{\text{next}}) = P_0(W_{\text{next}}) \cup \{i^*\} \setminus \{j^*\} \), which is our swapping idea in Step 3(b)iii. For any other item \( i \notin \{i^*, j^*\} \), \( x^*_i(W) \) does not change, and hence the profile stays the same for any \( W \in (W_{\text{next}}, \hat{W}_{\text{next}}) \), which is \( P(W) = \{P_1 \setminus \{i^*\}, (i^*, j^*), P_0 \setminus \{j^*\}\} \). By Lemma 1, this allows us to consider the same candidate set(s) for the interval \([W_{\text{next}}, \hat{W}_{\text{next}}])

![Figure 1](image-url)

**Figure 1** An illustration of how the optimal basic solution \( x^*(W) \) to Problem \( \text{KP-RELAX}(W, K) \) changes as \( W \) increases from \( W_{\text{next}} \) to \( \hat{W}_{\text{next}} \). Note that \( x_{i^*}(W) \) decreases until it hits zero at \( \hat{W}_{\text{next}} \), while \( x_{j^*}(W) \) increases until it hits one at \( \hat{W}_{\text{next}} \).
Theorem 5 (1/2 Approximation Algorithm). For any $z \geq 0$, the set $S_{1/2}$ returned by Algorithm 1 satisfies

$$\text{ms-cost}(S_{1/2}, z) \geq \frac{1}{2} \max_{W \in [0, \infty)} \text{KP}(W, K)$$

In addition, the overall complexity of Algorithm 1 is in the order of $O(n^2 (\log n + K))$.

We conclude Section 5.2 with a remark that as long as we have a 1/2-approx algorithm for Problem (sub-dual($z, K$)), one can then design a PTAS for Problem (sub-dual($z, K$)) that has a similar flavor as the PTAS in Caprara et al. (2000). To do that, one iterates over all subsets $S$ with cardinality $|S| \leq \min\{\lceil 1/\epsilon - 2 \rceil, K\}$ and solves a sub-knapsack problem defined on $[n] \setminus S$ using our 1/2-approx. algorithm. The details of the PTAS are deferred to Appendix G.

5.3. FPTAS for Problem (SUB-DUAL($z, K$))

In Section 5.2, we designed a 1/2-approx. algorithm for Problem (sub-dual($z, K$)) with the help of our transformation and in particular, an infinite series of parameterized knapsack problems, i.e., $\max_{W \geq 0} \text{KP}(W, K)$. In this section, again by relying on our transformation, we present an FPTAS to Problem (sub-dual($z, K$)).

The FPTAS we propose draws inspiration from the FPTAS developed in Caprara et al. (2000). The authors propose an FPTAS for a single knapsack problem with cardinality constraint, which uses a DP scheme. In our problem, however, one main difficulty is that we need to solve an infinite number of parameterized knapsack problems with cardinality constraint, where parameter $W \in [0, \infty)$ determines the capacity of the knapsack and also influences the utilities of the items. This further leads to a number of new challenges: (i) it is unrealistic to apply the aforementioned DP approach to every $W \in [0, \infty)$ as there are infinite number of them; (ii) in the FPTAS for a single knapsack problem in in Caprara et al. (2000), the authors re-scale the utility of each item, based on an 1/2-approx. solution to (kp($W, K$)), and the accuracy parameter $\epsilon$. Now, it becomes unclear how we would re-scale the utility of each item with an infinite number of knapsack problems, since the 1/2-approx. solution to (kp($W, K$)) changes with $W$. In the following, we propose an FPTAS for $\max_{W \geq 0} \text{KP}(W, K)$, which tackles these challenges.

We start by partitioning the interval $[0, \infty)$—which $W$ falls into in Problem $\max_{W \geq 0} \text{KP}(W, K)$—into $O(n)$ sub-intervals. These sub-interval are chosen such that in each such sub-intervals $I$, the set of eligible items $E_I$ defined below

$$E_I = \{i : i \in [n], w_i \leq W_{\text{min}}, u_i(W) \geq 0 \text{ for any } W \in I\}$$

remain the same. That is, on each sub-interval $I$, the set of items with positive utilities and weight less than $W_{\text{min}}$ remain the same. Note that in the optimal solution to a knapsack problem, items with negative utilities will be ignored. In addition, items with weight greater than $W_{\text{min}}$ cannot
fit into any of knapsack in \((\text{KP}(W, K)), W \in [W_{\text{min}}, W_{\text{max}}]\). Note that there are \(O(n)\) sub-intervals because the set of items that can fit into the knapsack changes at \(\{w_i : i \in [n]\}\), and the sign of each item changes at \(\{\frac{w_i}{c_i} - 1 : i \in [n]\}\). (Recall that \(u_i(W) = \frac{w_i}{1+W}-c_i\).) Hence, there are at most \(O(n)\) values of \(W\) at which the set of eligible items \(E_i\) changes. We next focus on each sub-interval \(I = [W_{\text{min}}, W_{\text{max}}]\) and design an FPTAS for \(\max_{W \in I} \text{KP}(W, K)\). That is, we find an assortment \(S_I\) with \(|S_I| \leq K\) such that \(\text{MS-Cost}(S_I) \geq (1 - \epsilon) \max_{W \in I} \text{KP}(W, K)\).

We first give an overview of our approach. As stated earlier, as our main hurdle, we need to solve an infinite series of parameterized knapsack problems \((\text{KP}(W, K)), W \in I = [W_{\text{min}}, W_{\text{max}}]\). To address this hurdle, we first solve the knapsack problem at capacity level \(W = W_{\text{min}}\) (i.e., \((\text{KP}(W_{\text{min}}, K))\)) via a DP scheme that we will describe shortly. As an important step of the DP scheme, we re-scale the utility of eligible items using a factor of \(K/(U \epsilon)\), where \(U = \max_{i \in E_i} u_i(W_{\text{min}})\) is the largest utility of a item at \(W_{\text{min}}\) and \(\epsilon\) is the accuracy parameter of FPTAS. This scaling is different from the re-scaling done in Caprara et al. (2000) that uses the \(1/2\)-approx. solution of \((\text{KP}(W_{\text{min}}, K))\). While this difference seems negligible at the first glance, it indeed plays a crucial role in our design. At a high level, we choose a re-scaling factor of the right size such that it simultaneously ensures (i) we can perform the DP scheme within reasonable running time (i.e., polynomial in \(n, K\) and \(1/\epsilon\)) and (ii) among the collection of sets we obtained during the run of the DP scheme, there exists a near-optimal set for the problem \(\max_{W \in I} \text{KP}(W, K)\).

We now describe our DP scheme for knapsack problem \(\text{KP}(W_{\text{min}}, K)\). In our DP scheme, we will not consider any of the ineligible items \(i \notin E_I\), as they either do not fit into the knapsack or would only reduce the utilities of an assortment. As mentioned before, we start by re-scaling the utility of each eligible item with a factor of \(K/(U \epsilon)\), where \(U = \max_{i \in E_i} u_i(W_{\text{min}})\). The re-scaled utility of item \(i \in E_I\), denoted by \(q_i\), is given by

\[
q_i = \left\lfloor \frac{u_i(W_{\text{min}})}{U \epsilon} K \right\rfloor.
\]

Note that after re-scaling, all eligible items have integer utilities, and since for any \(|S| \leq K\), we have \(\sum_{i \in S} u_i(W_{\text{min}}) \leq KU\), the optimal value to the knapsack problem with re-scaled utilities is bounded by \(KU \cdot \frac{K^2}{U\epsilon} + K \leq \left\lfloor \frac{K^2}{\epsilon} \right\rfloor + K := a_{\text{max}}\).

Having defined the re-scaled utilities, we proceed with the following DP recursion. While explaining the recursion, we also describe how we maintain a collection of assortments that contain an \((1 - \epsilon)\)-approximate solution to \(\max_{W \in I} \text{KP}(W, K)\), which involves an infinite number of knapsack problems. For any \(a \in \{0, 1, \ldots, a_{\text{max}}\}, b \in \{0, 1, \ldots, K\}, \) and \(i \in [n]\), let \(\text{MIN-WT}_i(a, b)\) denote the minimum total weight of \(b\) eligible items from \(\{1, \ldots, i\}\) that can yield total utility \(a\), and let \(\text{SET}_i(a, b)\) be the corresponding assortment consisting of these \(b\) items. (Here, \(\text{MIN-WT}\) stands for
“minimum weight.”) If we cannot find exactly \( b \) eligible items from \( \{1, \ldots, i\} \) whose total utility is \( a \), we set \( \text{MIN-WT}_i(a, b) = \infty \). More formally, we can define \( \text{MIN-WT}_i(a, b) \) as the following:

\[
\text{MIN-WT}_i(a, b) := \inf_{S \subseteq [i] \cap E_t} \left\{ \sum_{i \in S} w_i : \sum_{i \in S} q_i = a; \ |S| = b \right\}.
\]

If we were to solve a single knapsack problem \( \text{KP}(W_{\text{min}}, K) \), a near-optimal solution value would be given by

\[
\max_{a \in \{0, \ldots, a_{\text{max}}\}, b \in \{0, \ldots, K\}} \left\{ a : \text{MIN-WT}_n(a, b) \leq W_{\text{min}} \right\}.
\]

In our problem where there is an infinite number of knapsack problems, we instead consider all assortments \( \text{SET}_n(a, b) \) such that \( \text{MIN-WT}_n(a, b) \leq W_{\text{max}} \). That is, we consider all candidate assortments that can fit into a knapsack of capacity \( W_{\text{max}} \). Interestingly, we show that this collection of assortments contains a \((1 - \epsilon)\)-approximate solution to problem \( \max_{W \in E_t} \text{KP}(W, K) \). Among these assortments, we then choose the assortment \( S_I \) that maximizes the cost-adjusted market share (i.e., \( \text{MS-COST}(S, z) \)). We summarize the details of our FPTAS in Algorithm 2.

As for the recursion, initially, we set \( \text{MIN-WT}_0(0, 0) = 0 \) and \( \text{SET}_0(0, 0) = \emptyset \). We also let \( \text{MIN-WT}_0(a, b) = \infty \) for all \( a \in \{0, \ldots, a_{\text{max}}\} \), and \( b \in \{0, \ldots, K\} \). Finally, for any \( i \in \{0, 1, \ldots, n\} \), we set \( \text{MIN-WT}_i(a, b) = \infty \) and \( \text{SET}_i(a, b) = \emptyset \) when either \( a \) or \( b \) are negative. We can now compute the values for \( \text{MIN-WT}_i(.) \) using values from \( \text{MIN-WT}_{i-1}(.) \), and construct the assortment \( \text{SET}_i(.) \) from \( \text{SET}_{i-1}(.) \) in a recursive manner, for any eligible item \( i \in E_t \):

\[
\text{MIN-WT}_i(a, b) = \min \left\{ \text{MIN-WT}_{i-1}(a, b), \text{MIN-WT}_{i-1}(a - q_i, b - 1) + w_i \right\}
\]

Note that in the first term (i.e., \( \text{MIN-WT}_{i-1}(a, b) \)), we do not include item \( i \) in the set and in the second term (i.e., \( \text{MIN-WT}_{i-1}(a - q_i, b - 1) + w_i \)), we include item \( i \) in the set. If \( \text{MIN-WT}_i(a, b) < \infty \), we update its corresponding assortment:\(^{10}\)

\[
\text{SET}_i(a, b) = \begin{cases} \text{SET}_{i-1}(a, b) & \text{if } \text{MIN-WT}_i(a, b) = \text{MIN-WT}_{i-1}(a, b) \\ \text{SET}_{i-1}(a - q_i, b - 1) \cup \{i\} & \text{if } \text{MIN-WT}_i(a, b) = \text{MIN-WT}_{i-1}(a - q_i, b - 1) + w_i \end{cases}
\]

This then gives us access to the matrix \( \text{MIN-WT}_n(a, b) \). We note that if \( \text{MIN-WT}_n(a, b) < \infty \) for some \( (a, b) \), this means that \( \text{SET}_n(a, b) \) is the assortment with minimum weight that satisfies

\[
\sum_{i \in \text{SET}_n(a, b)} q_i = a \quad \text{and} \quad |\text{SET}_n(a, b)| = b.
\]

In the following theorem, which is the main result of Section 5.3, we show that the assortment \( S_I \) returned by Algorithm 2 is indeed near-optimal, and runs in time polynomial in \( n, K \) and \( 1/\epsilon \).

---

9 For ineligible item \( i \notin E_t \), we will simply let \( \text{MIN-WT}_i(a, b) = \text{MIN-WT}_{i-1}(a, b) \) and \( \text{SET}_i(a, b) = \text{SET}_{i-1}(a, b) \) for all \( a \geq a_{\text{max}}, b \leq K \). In this way, we ensure that none of the ineligible items would be included in our assortments.

10 When \( \text{MIN-WT}_i(a, b) = \infty \), its corresponding set is not used by our algorithm, and hence we do not define it.
Algorithm 2 FPTAS for $\max_{W \in I} \text{KP}(W, K)$

**Input:** weights $w = (w_1, \ldots, w_n)$, costs $c = (c_1, \ldots, c_n)$, cardinality upper bound $K$, interval $I = [W_{\min}, W_{\max}) \subset [0, \infty)$, eligible set $E_I$, and accuracy parameter $\epsilon \in (0, 1)$.

**Output:** assortment $S_I$

1. **Re-scale the utilities.** Let $U = \max_{i \in E_I} u_i(W_{\min})$, where $E_I$ is the set of eligible items and is defined in Equation (6). Re-scale the utility of each eligible item to be

   $$q_i = \left[ \frac{u_i(W_{\min})K}{U \epsilon} \right] \quad i \in E_I.$$ 

2. **Initialization.** Let $\text{MIN-WT}_0(0, 0) = 0$ and $\text{SET}_0(0, 0) = \emptyset$. Further, let $\text{MIN-WT}_0(a, b) = \infty$ for all $a \in \{0, \ldots, a_{\max}\}$, and $b \in \{0, \ldots, K\}$. Finally, for any $i \in \{0, 1, \ldots, n\}$, set $\text{MIN-WT}_i(a, b) = \infty$ and $\text{SET}_i(a, b) = \emptyset$ when either $a$ or $b$ are negative. Let $a_{\max} = \lceil \frac{K^2}{\epsilon} \rceil + K$. Initialize the collection of assortments $C_I = \emptyset$.

3. **Dynamic Programming.** For $i = 1, \ldots, n$,
   - If $i \notin E_I$, let $\text{MIN-WT}_i(a, b) = \text{MIN-WT}_{i-1}(a, b)$ and $\text{SET}_i(a, b) = \text{SET}_{i-1}(a, b)$ for all $a \leq a_{\max}, b \leq K$. Then, go to the start of Step (3) and begin the next iteration with $i + 1$.
   - For any $a \in \{0, \ldots, a_{\max}\}$ and $b \in \{0, \ldots, K\}$, compute the entries of $\text{MIN-WT}_i(.)$ as follows:

     $$\text{MIN-WT}_i(a, b) = \min \left\{ \text{MIN-WT}_{i-1}(a, b), \text{MIN-WT}_{i-1}(a - q_i, b - 1) + w_i \right\}$$

   - If $\text{MIN-WT}_i(a, b) < \infty$, update its corresponding assortment:

     $$\text{SET}_i(a, b) = \begin{cases} \text{SET}_{i-1}(a, b) & \text{if } \text{MIN-WT}_i(a, b) = \text{MIN-WT}_{i-1}(a, b) \\ \text{SET}_{i-1}(a - q_i, b - 1) \cup \{i\} & \text{if } \text{MIN-WT}_i(a, b) = \text{MIN-WT}_{i-1}(a - q_i, b - 1) + w_i \end{cases}$$

4. **Collect assortments.** For any $a \in \{0, \ldots, a_{\max}\}$ and $b \in \{0, \ldots, K\}$ such that $\text{MIN-WT}_n(a, b) \leq W_{\max}$, add $\text{SET}_n(a, b)$ to $C_I$.

5. Return the assortment $S_I = \arg \max_{S \in C_I} \text{MS-COST}(S, z)$.

**Theorem 6 (Near-optimality of Algorithm 2).** Consider any interval $I$ such that on this interval the set of eligible items $E_I$ (defined in Equation (6)) remains the same. Then, for any $z \geq 0$, set $S_I$, returned by Algorithm 2, has the following property

$$\text{MS-COST}(S_I, z) \geq (1 - \epsilon) \max_{W \in I} \text{KP}(W, K).$$

In addition, the overall complexity of Algorithm 2 is in the order of $O(nK^3/\epsilon)$.

Theorem 6 suggest how to design an FPTAS for the problem of $\max_{W \geq 0} \text{KP}(W, K)$. Recall that to solve this problem, we start by partitioning $[0, \infty)$ into $O(n)$ sub-intervals, and we then apply Algorithm 2 to each of the sub-intervals. Let $I$ denote the collection of these sub-intervals. Among
the near-optimal assortments we obtained for each sub-interval, we select the one with the highest cost-adjusted market share: \( S = \arg \max_{S' \in \{S_I : I \in \mathcal{I}\}} \text{MS-COST}(S', z) \). This assortment \( S \) then satisfies \( \text{MS-COST}(S, z) \geq (1 - \epsilon) \max_{W \in [0, \infty)} \text{KP}(W, K) \), and is thus a \((1 - \epsilon)\)-approximate solution to Problem \((\text{SUB-DUAL}(z, K))\). Theorem 6 further implies solving \( \max_{W \geq 0} \text{KP}(W, K) \) via Algorithm 2 has a total complexity of \( O(n^2 K^3 / \epsilon) \). This is because, as stated earlier, to solve \( \max_{W \geq 0} \text{KP}(W, K) \), we need to apply Algorithm 2 to \( O(n) \) sub-intervals, and the running time of Algorithm 2 on each sub-interval is at most \( O(n K^3 / \epsilon) \). Hence, by applying Algorithm 2 on \( O(n) \) sub-intervals, we have an FPTAS for solving problem \( \max_{W \geq 0} \text{KP}(W, K) \).

6. Numerical Studies

In this section, we numerically evaluate the performance of our 1/2-approx. fair Ellipsoid-based algorithm and FPTAS fair Ellipsoid-based algorithm, designed for Problem \((\text{FAIR})\), on both real-world and synthetic data. In Section 6.1, we evaluate our algorithms in a movie recommendation setting, where the popularity weight of each movie is based on the average rating from the MovieLens 100K (ML-100K) dataset (see Harper and Konstan (2015)). In Section 6.2, we synthetically generate the popularity weights of the items in a real-world inspired setting, where only a small number of items have much higher popularity, whereas the majority of the items are much less popular. For each instance of the assortment planning problem that we consider, we apply the two versions of our fair Ellipsoid-based algorithm—the 1/2-approx. algorithm and the FPTAS—to solving Problem \((\text{FAIR})\).

6.1. Experiments on MovieLens Dataset

**MovieLens dataset.** We design the following experiments using the MovieLens 100K (ML-100K) dataset, which contains 100,000 movie ratings, which take value in \{1, 2, 3, 4, 5\}, from 943 users on 1682 movies. The ratings were collected through the MovieLens website during September 1997 to April 1998. The movies belong to 19 different genres (e.g., action, drama, horror). In this example, we act as a movie recommendation platform that needs to feature an assortment of movies of a particular genre.

**Setup.** In the following experiments, we focus on the subset of \( n = 20 \) drama movies\(^{12}\) that have been rated by at least five users and have an average rating of at least 3. Let \( r_i \) denote the average

---

\(^{11}\)In our implementation of the fair Ellipsoid-based algorithm, we stop the Ellipsoid method under one of the following two scenarios: (i) The Ellipsoid method terminates after it reduces the ellipsoid to a sufficiently small size. (ii) At the end of each iteration of the Ellipsoid method (Algorithm 3), we use the sets from the collection \( \mathcal{V} \) to solve the primal problem \((\text{FAIR})\). If the market share obtained at two subsequent iterations has a difference less than 0.001, we terminate the fair Ellipsoid-based algorithm.

\(^{12}\)We have performed the same experiments on movies of different genres and the results are consistent across genres.
rating of the $i$th drama movie, for $i \in [20]$. Without loss of generality, we let $r_1 \geq r_2 \geq \cdots \geq r_{20}$. Among them, the highest average rating is $r_1 = 4.09$ and the lowest is $r_{20} = 3.14$. We let the popularity weight of each movie be proportional to their average rating: $w_i = s \cdot r_i$. In the following, we take the scaling factor to be $s = 1/20$.\textsuperscript{13} We consider an assortment planning problem in a movie recommendation setting, where we can only recommend at most 5 out of these 20 drama movies (that is, we can only place $K = 5$ movies in our offered assortment). Under this setting, there are a total of $\sum_{k=1}^{5} \binom{20}{k} = 21,699$ different assortments that to randomize over. We apply the $1/2$-approx. and FPTAS fair Ellipsoid-based algorithm to obtaining near-optimal solutions for this problem.

**Price of fairness.** We first investigate the price introduced by the fairness constraints, which can be seen in the following two aspects shown in Figure 2.

![Figure 2](image_url)

(a) The market share obtained by our algorithm under different levels of fairness constraints. The dashed line shows the optimal market share in the absence of fairness constraints.

(b) The number of sets that our algorithms need to randomize over under different levels of fairness constraints.

**Figure 2** The price introduced by fairness constraints in the MovieLens dataset.

1. **Impact on market share.** Figure 2a shows the market share (in this example, the probability of a successful movie recommendation) obtained by our algorithms under different levels of fairness constraints (i.e., $\delta$). The dashed line in this figure shows the average optimal market share in the absence of any fairness constraints, and hence it can be used to measure the price of fairness. As expected, we observe a clear trade-off between fairness and market share. As the fairness constraint gets relaxed (i.e., as $\delta$ increases), the market share increases and approaches the maximum market share attainable in the absence of fairness. That being said, even when

\textsuperscript{13}Note that the scaling factor $s$ determines how each movie is weighted against the no-purchase option (i.e., a user chooses not to watch any movie). We obtain consistent results when we set the scaling factor to be $s = 1/10$. Additionally, note that when we set $s = 1/20$, the weights of the movies $\{w_i\}_{i \in [20]}$ are very close to each other, and hence it becomes crucial to enforce fairness constraints.
we require the platform to be 0-fair \((\delta = 0)\), the price of fairness is quite small, as the average market share only gets reduced by around 4\% in our near-optimal solutions. This implies that we can achieve a high level of fairness in the movie recommendation platform without exerting heavy impact on the market share.

In terms of our choice of the algorithm for Problem \((\text{sub-dual}(z, K))\), although theoretically, the FPTAS has a better approximation factor, the performance of the 1/2-approx. algorithm and the FPTAS are in fact quite comparable under all levels of fairness constraints. As \(\delta\) gets larger, the 1/2-approx. algorithm even slightly outperforms the FPTAS. This is because in our example, the popularity weights of the 20 movies are very close to each other\(^{14}\). Due to the re-scaling mechanism of the FPTAS, if the weights are close to each other, the movies might be indistinguishable by their scaled utilities, and the FPTAS tends to prioritize movies with smaller weights when it constructs the assortments with minimum weight set \(\hat{n}(.)\) in the DP scheme (see Section 5.3). This then can result in lower market share.

2. Impact on the number of randomized sets. The introduction of fairness constraint also requires us to randomize over a number of sets instead of sticking with a fixed assortment. Figure 2b shows the number of sets\(^{15}\) that we need to randomize over in our near-optimal solutions, under different levels of fairness constraints. We observe that the number of sets ranges from 1 to 20, highlighting our algorithms can be effectively implemented in practice. Under our near-optimal solutions, we only need to randomize over at most \(n = 20\) sets even when \(\delta = 0\). Intuitively, this is because, in our algorithms, each set that we consider allows us to satisfy the fairness constraint for one more movie. As another observation, as we relax the fairness constraints and increase \(\delta\), the number of sets we need to randomize over decreases. Finally, note that the performance of our 1/2-approx. algorithm is again comparable to that of our FPTAS.

Visibility. We next investigate the visibility of each movie (i.e., the probability to be included in the offered assortment) under different levels of fairness constraints. Recall that we have ranked the movies in the descending order of their weights: \(w_1 \geq w_2 \geq \cdots \geq w_{20}\). Figure 3 depicts the visibility of each movie under our 1/2-approx. fair Ellipsoid-based algorithm with different choices of \(\delta\).\(^{16}\) In this figure, we also include the visibility of each movie when there is no fairness constraint. We observe that the visibility of movies decreases as their weights get smaller. Further, in our algorithm, for each \(\delta\) there is a threshold such any movie with an index greater than this threshold

\(^{14}\) In our synthetic experiments, we investigate a different scenario where one item has a much larger weight than the rest of the items; see Section 6.2.

\(^{15}\) Here we only consider sets \(S\) with \(\hat{p}(S) > 10^{-3}\) in the near-optimal solution \(\hat{p}(.)\).

\(^{16}\) Here we focus on the performance of our 1/2-approx. algorithm as it is comparable with that of the FPTAS.
gets positive visibility while the rest of movies get zero visibility. Note that this threshold gets smaller as \( \delta \) increases, implying that the parameter \( \delta \) can be effectively tuned to strike a balance between fairness (i.e., visibility of low-weight items) and the induced market share.

![Figure 3](image)

**Figure 3** The visibility of each movie \( i \in [20] \) under different levels of fairness constraints (i.e., \( \delta \)).

**Runtime Comparison.** Finally, we compare the computational performance of our 1/2-approx. and FPTAS fair Ellipsoid-based algorithm. In Table 1, we record the runtime taken by these two algorithms. Here, the time taken by a single execution refers to the time taken to solve one instance of Problem \((\text{sub-dual}(z, K))\). Recall from Section 4.3 that we need to solve many instances of Problem \((\text{sub-dual}(z, K))\) throughout the run of the Ellipsoid method; here we also record the total time taken by our 1/2-approx. and FPTAS when solving one instance of Problem \((\text{fair})\) using the fair Ellipsoid-based algorithm. We observe that the 1/2-approx. algorithm is much faster than the FPTAS, in terms of both the time needed for one execution, and the total time taken in solving Problem \((\text{fair})\). This makes the 1/2-approx. algorithm a very appealing practical option.

| \( \delta \) | 1/2-approx | FPTAS |
| --- | --- | --- |
| 0 | 1.6 \( \times 10^{-3} \) | 3.0 \( \times 10^{-1} \) |
| 0.2 | 1.5 \( \times 10^{-3} \) | 3.0 \( \times 10^{-1} \) |
| 0.4 | 1.4 \( \times 10^{-3} \) | 3.1 \( \times 10^{-1} \) |
| 0.6 | 1.5 \( \times 10^{-3} \) | 2.8 \( \times 10^{-1} \) |
| 0.8 | 1.5 \( \times 10^{-3} \) | 3.1 \( \times 10^{-1} \) |
| 1.0 | 1.5 \( \times 10^{-3} \) | 3.1 \( \times 10^{-1} \) |

We can connect the sharp difference in runtime back to the theoretical complexity of these two algorithms in the worst case. We show in Theorem 5 that the 1/2-approx. algorithm for Problem \((\text{sub-dual}(z, K))\) takes at most \( O(n^2(\log n + K)) \) time, and in Theorem 6 that the FPTAS for Problem \((\text{sub-dual}(z, K))\) takes at most \( O(n^2 K^3/\epsilon) \) time. Observe that (i) in terms of dependency on \( K \), the worst-case complexity of the 1/2-approx. algorithm is better than that of FPTAS, and (ii) the running time of the FPTAS is additionally polynomial in \( 1/\epsilon \). These worst-case results, however, may be far from the running time of these algorithms in realistic scenarios, as we have
observed in Table 1. The table shows that the gap between the running time of the aforementioned algorithms could be even larger than what the theory suggests. To see why, recall that the most time-consuming step of the 1/2-approx. algorithm is the dynamic partitioning of the interval $I_{\text{high}}$. This interval can be partitioned into at most $O(nK)$ sub-intervals, where we need to solve some simple optimization problem for each sub-interval. In our instance, however, the number of sub-intervals constructed in 1/2 approx. algorithm is quite small, allowing the algorithm to have a much better running time than its worst-case bound. In our FPTAS, on the other hand, each execution always requires us to first divide $[0, \infty)$ into $n$ sub-intervals and then construct a matrix of size $O(nK^2/\epsilon)$, and hence there is very little room for improvement.

6.2. Experiments on Synthetic Data

In this section, we verify our findings in previous numerical studies on the MovieLens dataset by reproducing the experiments on a number of synthetically generated datasets.

**Setup.** We consider an assortment planning problem with $n = 20$ items, where the cardinality of the offered assortment is at most $K = 5$. We generate 100 instances, where in every instance, $w_1$ is drawn from a uniform distribution in $[0.2, 0.5]$, and $w_i$, $i \in \{2, \ldots, 20\}$, is drawn from a uniform distribution in $[0, 0.1]$. Observe that in this setting, one item has a relatively high weight and the rest of the items have a smaller weight. This is motivated by realistic scenarios under which there usually exist a small number of items that are most popular (e.g., the best sellers), while the majority of the items are of less quality and popularity. Again, without loss of generality, we re-index the items such that $w_1 \geq w_2 \geq \cdots \geq w_n$.

**Price of fairness.** In Figure 4, we show the impact of fairness constraints on two aspects—the market share and the number of randomized sets—in our near-optimal solutions. Similar to what we have seen in the experiments on MovieLens dataset, the performance of the 1/2-approx. algorithm and the FPTAS are quite comparable in both aspects. In Figure 4a, we again observe a trade-off between fairness and market share. We note that when $\delta = 0$, the price of fairness is still quite small, as the average market share only gets reduced by around 15% in our near-optimal solutions. Compared to the movie recommendation example, imposing fairness constraints reduce our market share to a greater extent. This is because our setup in the synthetic experiments makes satisfying the fairness constraints more difficult. In this setup, the weights of the items are no longer close to each other, and therefore we need to provide sufficient visibility to the item with high weight, and meanwhile, maintain high enough visibility for the rest of the items. In Figure 4b, we observe that as $\delta$ increases, the average number of sets we need to randomize over decreases and the variance in the number of sets increases. This again implies that there are more chances to consider a much smaller number of sets as fairness constraints relax.
Figure 4  The price introduced by fairness constraints in the synthetic dataset.

Runtime comparison. The average runtime of our 1/2-approx. algorithm and FPTAS on our synthetically generated instances (recorded in Table 2) is consistent with what we have seen in Table 1, which demonstrates the fast computational performance of the 1/2-approx. algorithm. Note that in Table 2, the total time taken in solving Problem (FAIR) is longer for both algorithms, compared to the movie recommendation setting. This is again due to the additional difficulty introduced by our setup in the synthetic experiments. The problem is especially time-consuming when δ is small, which reduces the feasibility region of Problem (FAIR) even further and makes our search for a feasible solution more difficult.

| Table 2 Average runtime comparison of 1/2-approx. and FPTAS algorithms. |

|                      | δ    | 0       | 0.2     | 0.4     | 0.6     | 0.8     | 1.0     |
|----------------------|------|---------|---------|---------|---------|---------|---------|
| Time taken by a      |      |         |         |         |         |         |         |
| single execution (s) | 1/2-approx | 8.3 × 10⁻⁴ | 8.3 × 10⁻⁴ | 8.4 × 10⁻⁴ | 9.2 × 10⁻⁴ | 1.0 × 10⁻³ | 9.0 × 10⁻⁴ |
|                      | FPTAS | 3.1 × 10⁻¹ | 3.0 × 10⁻¹ | 3.1 × 10⁻¹ | 3.2 × 10⁻¹ | 3.1 × 10⁻¹ | 3.1 × 10⁻¹ |
| Total time taken in   |      |         |         |         |         |         |         |
| solving (FAIR) (s)   | 1/2-approx | 47.4     | 36.0     | 25.0     | 10.4     | 4.4     | 2.3     |
|                      | FPTAS | 6571.3   | 2892.7   | 697.7    | 1004.5   | 696.8   | 677.6   |

7. Extensions and Discussions

In this section, we consider a few extensions to the problem setup in Section 3 and discuss how our approaches can be extended to the more general settings where each item comes with a profit margin and when different notions of fairness are considered.
7.1. Assortment Planning with Profits

We consider the more general setting where the sale of each item $i$ generates profit $r_i > 0$, and the expected revenue of offering assortment $S$ is $\text{REV}(S) = \sum_{i \in S} \frac{r_i w_i}{1 + \sum_{i \in S} w_i}$. Under this setting, the platform’s objective is to maximize the expected revenue of its randomized assortments, i.e.,

$$\max_{p(S) \geq 0, |S| \leq K} \sum_{S: |S| \leq K} p(S) \cdot \text{REV}(S),$$

with the same set of constraints as in Problem (FAIR).

Our Ellipsoid-based framework can naturally extend to this setting, since this only changes the dual fairness constraint in Problem (FAIR-DUAL) slightly to

$$\sum_{i \in S} \left( \sum_{j=1, j \neq i}^{n} z_{ij} w_j - \sum_{j=1, j \neq i}^{n} z_{ji} w_j \right) + \rho \geq \text{REV}(S) \quad \forall S: |S| \leq K$$

Hence, we can simply reformulate Problem (SUB-DUAL($z, K$)) to be

$$\text{SUB-DUAL}(z, K) = \max_{S: |S| \leq K} \left\{ \text{REV}(S) - \sum_{i \in S} c_i(z) \right\}. \quad (8)$$

Using the same arguments as in Theorem 3, we will obtain a $\beta$-approximation algorithm for the more general fair assortment planning problem, as long as we can find a $\beta$-approximate set for Problem (SUB-DUAL($z, K$)) defined in (8). We can now propose near-optimal algorithms for the new Problem (SUB-DUAL($z, K$)) using similar ideas from Section 5.

As shown in Theorem 4, we first note that the new Problem (SUB-DUAL($z, K$)) is again equivalent to an infinite series of knapsack problems $\max_{W \geq 0} \text{KP}(W, K)$, where

$$\text{KP}(W, K) = \max_{S} \sum_{i \in S} \frac{r_i w_i}{1 + W} - c_i \quad \text{s.t.} \quad \sum_{i \in S} w_i \leq W \quad \text{and} \quad |S| \leq K. \quad (9)$$

Note that the parameterized knapsack problems are only different from the previous knapsack problems in the utility terms, which now become $u_i(W) = \frac{r_i w_i}{1 + W} - c_i$.

Using the transformation above, we can make the following modifications to our 1/2-approx. algorithm (presented in Algorithm 1) and obtain the same 1/2 approximation factor for the new problem. First, before applying the 1/2-approx. algorithm, we need to pre-partition the interval $[0, \infty)$ into $O(n^3)$ sub-intervals, such that on each sub-interval $I$, three conditions are satisfied:

1. The ordering of the utilities of items does not change. Note that the ordering of utility only changes at the values of $W$ such that $\frac{r_i w_i}{1 + W} - c_i = \frac{r_j w_j}{1 + W} - c_j$ for some $i \neq j$, and hence there are $O(n^2)$ such values of $W$ where the ordering changes.
(2) The ordering of the utility-to-weight ratios of items does not change. Note that the utility-to-weight ratio only changes at the values of $W$ such that $rac{r_i}{1 + W} - \frac{c_i}{w_i} = \frac{r_j}{1 + W} - \frac{c_j}{w_j}$ for some $i \neq j$, and there are again $O(n^2)$ such values of $W$ where the ordering changes. This condition ensures that Step 1b of Algorithm 1 is well-defined, since we need to rank the items by their utility-to-weight ratio.

For any three items $(i,j,k)$ such that $i \neq j, j \neq k, k \neq i$, the sign of the reduced cost of item $k$ with items $i,j$ as the basic variables does not change. Note that this sign only changes when $(w_i - w_j)u_k(W) + (w_j - w_k)u_i(W) + (w_k - w_i)u_j(W) = 0$, and there are only $O(n^3)$ such values of $W$ at which the sign changes. This ensures that when the capacity $W$ changes, the sign of the reduced cost of any item does not change as long as $W$ remains on sub-interval $I$. This condition is crucial for our proof of Lemma 7 and 8.

After pre-partitioning $[0, \infty)$ into $O(n^3)$ sub-intervals such that the three conditions above are met on each sub-interval $I$, we next apply our 1/2-approx. algorithm to each sub-interval $I$ separately, to obtain a 1/2-approx. solution for $\max_{W \in I} \text{KP}(W,K)$. To do that, we make slight modifications to the adaptive partitioning procedure in Step 3b of Algorithm 1, mainly in the following two aspects:

1. We modify the criterion used for identifying the two items $(i^{*},j^{*})$ to be swapped as follows:
   
   $$(i^{*},j^{*}) \leftarrow \arg \max_{i \in \mathcal{P}_1, j \in \mathcal{P}_0 \text{ where } w_i < w_j} \left[ \frac{u_j(W_{\text{next}}) - u_i(W_{\text{next}})}{w_j - w_i} \right],$$

   where $u_k(W) = \frac{r_k w_k}{1 + W} - c_k$ is the utility of item $k$. Here, we again choose two items such that swapping them yields the highest marginal increase in total utility, taking into account the profit $r_i$ of each item. Note that the new criterion accounts for the changes in utilities of items.

2. We also modify the stopping rule in Step 3b to be: if $u_{j^{*}}(W_{\text{next}}) \geq u_{i^{*}}(W_{\text{next}})$, we add $P_1$ to $\mathcal{C}$. Otherwise, we go to the termination step. That is, we have reached the end of the algorithm if we can no longer increase the total utility by swapping two items.

The modifications above allow us to find a polynomial-time 1/2-approx. solution for the new infinite series of knapsack problems $\max_{W \in [0, \infty)} \text{KP}(W,K)$. We provide the formal algorithm of the modified 1/2-approx. algorithm outlined above in Algorithm 5 (see Appendix I).

**Proposition 1 (1/2-Approx. Algorithm for Problem (SUB-DUAL($z,K$)) with Profits).**

Consider any interval $I$ such that on this interval (i) the ordering of utilities do not change; (ii) the ordering of utility-to-weight ratios do not change; (iii) the condition in (10) is satisfied. Then, for any $z \geq 0$, the set $S_{1/2}$ returned by Algorithm 5 satisfies

$$\text{ms-cost}(S_{1/2}, z) \geq \frac{1}{2} \max_{W \in I} \text{KP}(W,K),$$

where $\text{KP}(W,K)$ is the knapsack problems defined in (9). Given Algorithm 5, the overall complexity to find a 1/2-approx. solution for $\max_{W \in [0, \infty)} \text{KP}(W,K)$ is in the order of $O(n^5(\log n + K))$. 

Finally, recall that at the end of Section 5.2, we remark that with the help of our $1/2$-approx. algorithm, we can design a PTAS, which is presented in Algorithm 4 in Appendix G. Note that regardless of the setting we consider, as long as we have a $1/2$-approx algorithm for Problem (sub-dual$(z,K)$), one can always design a PTAS for Problem (sub-dual$(z,K)$), which takes the $1/2$-approx. algorithm as an input. In the more general setting where each item comes with a profit, the design of the PTAS is exactly the same as the one described in Appendix G, except that (i) the utility $u_i(W)$ of each item $i$ is now defined differently and depends on profit $r_i$; and (ii) it now takes the modified version of the $1/2$-approx. algorithm (Algorithm 5) as an input, which is used to solve the sub-knapsack problems in Step 3 of Algorithm 4. More formally, we have the following proposition, which extends results from Proposition 3.

**Proposition 2 (PTAS for Problem (SUB-DUAL$(z, K)$) with Profits).** Consider any interval $I$ such that on this interval the ordering of utilities of items do not change. Then, for any $z \geq 0$, set $S_I$, returned by the modified Algorithm 4, has the following property

$$\text{ms-cost}(S_I, z) \geq (1 - \epsilon) \max_{W \in I} \text{KP}(W, K)$$

In addition, its overall complexity is in the order of $O(n^{[1/\epsilon]+3}(\log n + K))$.

### 7.2. Different Notions of Fairness

As discussed in Section 3, a platform might wish to impose fairness with respect to some other quality metric $q_i(w_i, r_i)$ of item $i$, as stated in (2), instead of the popularity weight $w_i$. Some examples of the quality metric include $q_i = f(w_i)$, where $f$ is an increasing function; or $q_i = f(w_i) + g(r_i)$, where $f, g$ are both increasing functions. In general, we expect the quality of the item to be positively correlated with its popularity and the profit it gains, but we do not impose any assumptions to the quality function $q_i(w_i, r_i)$ used here.

We note that even when we impose fairness with respect to an arbitrary quality function $q_i(w_i, r_i)$, our fair Ellipsoid-based framework presented in Section 4.3 continues to work, as long as we modify our definition of cost of item $i$ to be

$$c_i(z) = \sum_{j=1, j \neq i}^{n} z_{ij}q_j(w_j, r_j) - \sum_{j=1, j \neq i}^{n} z_{ji}q_j(w_j, r_j)$$

The near-optimal algorithms for Problem (sub-dual$(z,K)$) presented in Section 5 (with the objective of maximizing market share) and in Section 7.1 (with the objective of maximizing expected revenue) both work for any fixed costs and do not rely on the expression of $c_i(z)$. Our theoretical results thus carry over naturally to a variety of fairness notions.
8. Conclusion and Future Directions

Many online platforms nowadays have largely relied on algorithms to optimize their operational decisions. However, in a variety of online platforms we have discussed, the use of single-minded algorithms that are solely concerned about maximizing the platform’s market share would only prioritize items that are most popular, thus leading to an undesirable “winner-take-all” phenomenon where the rest of the items face unfair outcomes. Motivated by this, we introduce the fair assortment planning problem (Problem (fair)), which involves a novel pairwise fairness notion that ensures items with similar merits get similar visibility. While many previous works on online operational decisions have focused on maximizing the utilities of the users or the platform, our work provides a new perspective that takes the interest of items on online platforms into account. Here, the items can take a variety of forms based on the given contexts, including products on online retail stores, posts on social media, or job candidates on job search sites. This perspective is particularly important because an unfair allocation of user exposure would discourage many of the product sellers/contents creators/job candidates from remaining on the platform, which, as a result, will negatively impact the prospect of the platform and narrow down users’ choices.

At a high level, in our work, we propose a novel framework that would give us access to a wide variety of near-optimal algorithms for Problem (fair). Based on that, we develop two fair Ellipsoid-based algorithms—an 1/2-approx. algorithm and a FPTAS—that return near-optimal solutions to Problem (fair) in polynomial time. These solutions require us to randomize over a polynomial number of different assortments, which can be effectively implemented in practice. Theoretically, our FPTAS attains a better approximation ratio, but we have showed, via numerical studies, that the 1/2-approx. algorithm has better computational performance and returns solutions that are comparable to those of the FPTAS. Another crucial insight that we have gained from our case study is that in many real-world settings, we can maintain a high level of fairness in the platform at the expense of only a slight decrease in market share as well as randomization over a small number of assortments. This implies that imposing fairness in online marketplaces can potentially be done at very little cost to the platform; however, such practice would improve satisfaction among the sellers and benefit the platform in the long run.

Our work opens up a number of new directions for future research. One natural direction is to further investigate the extensions proposed in Section 7. As discussed, our Ellipsoid-based framework and our approximate separation oracles used in this framework can be applied to different objective functions and any pairwise notion of fairness. It will be of interest to further explore how our unified framework works under other settings (e.g. different choice models and fairness notions). Another question of interest is to study fair assortment planning problems in a dynamic setting. In our work, we allocate visibility to each item in a single shot based on their popularity.
weights. It would be interesting to extend our model to consider dynamic settings under which
the quality or popularity of items evolve over time and needs to be continuously learned by the
algorithm. In this case, a new notion of fairness also needs to be proposed to ensure that we always
allocate visibility in a fair manner as we learn the most updated weights. In sum, fairness has only
been scarcely studied in operational decisions of online platforms, and we hope this work would
inspire a new stream of research in this area.

References
Jackie Baek and Vivek F Farias. Fair exploration via axiomatic bargaining. arXiv preprint arXiv:2106.02553,
2021.
Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. Regularized online allocation problems: Fairness and
beyond. In International Conference on Machine Learning, pages 630–639. PMLR, 2021.
Mohammad Hossein Bateni, Yiwei Chen, Dragos Florin Ciocan, and Vahab Mirrokni. Fair resource allocation
in a volatile marketplace. Operations Research, 2021.
Dimitris Bertsimas and John N Tsitsiklis. Introduction to linear optimization. Athena Scientific, 1997.
Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. The price of fairness. Operations research, 59
(1):17–31, 2011.
Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. On the efficiency-fairness trade-off. Management
Science, 58(12):2234–2250, 2012.
Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong,
Ed H Chi, et al. Fairness in recommendation ranking through pairwise comparisons. In Proceedings of the 25th
ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2212–2220, 2019.
Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness
in rankings. In The 41st international acm sigir conference on research & development in information
retrieval, pages 405–414, 2018.
Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Submodular maximization with
cardinality constraints. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete
Algorithms, SODA ’14, page 1433–1452, USA, 2014. Society for Industrial and Applied Mathematics.
Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building classifiers with independency constraints.
In 2009 IEEE International Conference on Data Mining Workshops, pages 13–18. IEEE, 2009.
Alberto Caprara, Hans Kellerer, Ulrich Pferschy, and David Pisinger. Approximation algorithms for knapsack
problems with cardinality constraints. European Journal of Operational Research, 123(2):333–345,
2000. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(99)00261-1. URL https://www.
sciencedirect.com/science/article/pii/S0377221799002611.
Maxime C Cohen, Ilan Lobel, and Renato Paes Leme. Feature-based dynamic pricing. *Management Science*, 66(11):4921–4943, 2020.

Maxime C Cohen, Adam N Elmachtoub, and Xiao Lei. Price discrimination with fairness constraints. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 2–2, 2021.

James M Davis, Guillermo Gallego, and Huseyin Topaloglu. Assortment optimization under variants of the nested logit model. *Operations Research*, 62(2):250–273, 2014.

Mahsa Derakhshan, Negin Golrezaei, and Renato Paes Leme. Lp-based approximation for personalized reserve prices. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 589–589, 2019.

Mahsa Derakhshan, Negin Golrezaei, Vahideh Manshadi, and Vahab Mirrokni. Product ranking on online platforms. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 459–459, 2020.

Kate Donahue and Jon Kleinberg. Fairness and utilization in allocating resources with uncertain demand. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 658–668, 2020.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

Naman Goel, Mohammad Yaghini, and Boi Faltings. Non-discriminatory machine learning through convex fairness criteria. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Negin Golrezaei, Hamid Nazerzadeh, and Paat Rusmevichientong. Real-time optimization of personalized assortments. *Management Science*, 60(6):1532–1551, 2014.

Negin Golrezaei, Adel Javanmard, and Vahab Mirrokni. Dynamic incentive-aware learning: Robust pricing in contextual auctions. *Operations Research*, 69(1):297–314, 2021a.

Negin Golrezaei, Vahideh Manshadi, Jon Schneider, and Shreyas Sekar. Learning product rankings robust to fake users. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 560–561, 2021b.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29:3315–3323, 2016.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

Brian Hayes. Computing science: The easiest hard problem. *American Scientist*, 90(2):113–117, 2002.
John N Hooker and H Paul Williams. Combining equity and utilitarianism in a mathematical programming model. *Management Science*, 58(9):1682–1693, 2012.

Nathan Kallus and Angela Zhou. The fairness of risk scores beyond classification: Bipartite ranking and the xauc metric. *Advances in neural information processing systems*, 32:3438–3448, 2019.

Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*, pages 2564–2572. PMLR, 2018.

Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 100–109, 2019.

A Gurhan Kok, Marshall L Fisher, and Ramnath Vaidyanathan. Assortment planning: Review of literature and industry practice. *Retail supply chain management*, 122:99–153, 2008.

Caitlin Kuhlman, MaryAnn VanValkenburg, and Elke Rundensteiner. Fare: Diagnostics for fair ranking using pairwise error metrics. In *The World Wide Web Conference*, pages 2936–2942, 2019.

Sumit Kunnumkal, Paat Rusmevichientong, and Huseyin Topaloglu. Assortment optimization under multinomial logit model with product costs. 2010. URL https://drive.google.com/file/d/1fkGNk_7sPdo24T7wZAE51isRvm4vzjVR.

Matt J Kusner, Joshua R Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *arXiv preprint arXiv:1703.06856*, 2017.

Will Ma, Pan Xu, and Yifan Xu. Fairness maximization among offline agents in online-matching markets. *arXiv preprint arXiv:2109.08934*, 2021.

Vahideh Manshadi, Rad Niazadeh, and Scott Rodilitz. Fair dynamic rationing. *Available at SSRN 3775895*, 2021.

Justin Mulvany and Ramandeep S Randhawa. Fair scheduling of heterogeneous customer populations. *Available at SSRN 3803016*, 2021.

Harikrishna Narasimhan, Andrew Cotter, Maya Gupta, and Serena Wang. Pairwise fairness for ranking and regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5248–5255, 2020.

Rad Niazadeh, Negin Golrezaei, Joshua R Wang, Fransisca Susan, and Ashwinkumar Badanidiyuru. Online learning via offline greedy algorithms: Applications in market design and optimization. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 737–738, 2021.

Paat Rusmevichientong, Zuo-Jun Max Shen, and David B Shmoys. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research*, 58(6):1666–1680, 2010.
Search Engine Journal. Amazon’s search engine ranking algorithm: What marketers need to know. https://www.searchenginejournal.com/amazon-search-engine-ranking-algorithm-explained/, 2018. Accessed: 2022-05-22.

Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2219–2228, 2018.

Ashudeep Singh and Thorsten Joachims. Policy learning for fairness in ranking. In Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.

Ashudeep Singh, David Kempe, and Thorsten Joachims. Fairness in ranking under uncertainty. Advances in Neural Information Processing Systems, 34, 2021.

The New York Times. Food businesses lose faith in Instagram after algorithm changes. https://www.nytimes.com/2022/03/22/dining/instagram-algorithm-reels.html, 2022. Accessed: 2022-03-22.

Kenneth Train. Qualitative choice analysis: Theory, econometrics, and an application to automobile demand, volume 10. MIT press, 1986.

Berk Ustun, Yang Liu, and David Parkes. Fairness without harm: Decoupled classifiers with preference guarantees. In International Conference on Machine Learning, pages 6373–6382. PMLR, 2019.

VentureBeat. LinkedIn says it reduced bias in its connection suggestion algorithm. https://venturebeat.com/2021/08/05/linkedin-says-it-reduced-bias-in-its-connection-suggestion-algorithm/, 2021. Accessed: 2022-03-22.

Vox. Can monoculture survive the algorithm? https://www.vox.com/the-goods/2019/12/17/21024439/monoculture-algorithm-netflix-spotify, 2019. Accessed: 2022-03-24.

Ruxian Wang and Ozge Sahin. The impact of consumer search cost on assortment planning and pricing. Management Science, 64(8):3649–3666, 2018.

Indre Zliobaite. On the relation between accuracy and fairness in binary classification. arXiv preprint arXiv:1505.05723, 2015.
Appendix A: Proof of Theorem 1

Here, we restate Problem (FAIR) for reader’s convenience.

\[
\text{FAIR} = \max_{p \in [0,1]^n : |S| \leq K} \sum_{S : |S| \leq K} p(S) \cdot \text{MS}(S)
\]

\[
\text{s.t.} \quad \frac{w_j}{w_i} \sum_{S \in S} p(S) - \sum_{S \in S} p(S) \leq \delta \quad i, j \in [n], i \neq j
\]

\[
\sum_{S : |S| \leq K} p(S) \leq 1
\]

\[
p(S) \geq 0 \quad S \subseteq [n], |S| \leq K,
\]

We start by showing that Problem (FAIR) admits an optimal basic feasible solution (for definition of basic feasible solution, see Definition 2.9 in Bertsimas and Tsitsiklis (1997) or Appendix J). Consider the polyhedron defined by the feasibility region of Problem (FAIR). Note that this polyhedron is (i) nonempty, because \(p(S) = 0\) for all \(S \subseteq [n], |S| \leq K\) is a feasible solution for any \(\delta \geq 0\); (ii) bounded, because \(p(S) \in [0,1]\) for all \(S \subseteq [n], |S| \leq K\). Then, by Theorem 2.6 in Bertsimas and Tsitsiklis (1997), there exists at least one extreme point in the polyhedron (for definition of extreme point, see Definition 2.6 in Bertsimas and Tsitsiklis (1997)). Since we have \(\text{MS}(S) \leq 1\) for any \(S\), this implies that \(\sum_{S : |S| \leq K} p(S)\text{MS}(S) \leq \sum_{S : |S| \leq K} p(S) \leq 1\); that is, the objective of Problem (FAIR) is also bounded. By Theorem 2.8 in Bertsimas and Tsitsiklis (1997), since the polyhedron has at least one extreme point and Problem (FAIR) has a bounded optimal objective, there must exist one extreme point of the polyhedron that is optimal. By Theorem 2.3 in Bertsimas and Tsitsiklis (1997), we have that an extreme point is equivalent to a basic feasible solution. Hence, we must have a basic feasible solution to Problem (FAIR) that is optimal, which we denote by \(p^*(.)\).

Now, let \(N = |\{S \subseteq [n] : |S| \leq K\}|\) be the number of variables in Problem (FAIR). There are \(n(n-1) + N\) constraints in Problem (FAIR), where \(n(n-1)\) of them are in the first set of constraints, and \(N\) of them are the non-negativity constraints. By definition of a basic feasible solution, we must have that at least \(N\) active (binding) constraints at \(p^*(.)\). That implies that at most \(n(n-1) + 1\) of the non-negativity constraints can be inactive, i.e. \(p^*(S) > 0\). We have thus showed that \(|\{S : p^*(S) > 0\}| \leq n(n-1) + 1\).

Appendix B: Details on the Ellipsoid Method

In this section, we provide more details on the Ellipsoid method used in Step 1 of the fair Ellipsoid-based algorithm, proposed in Section 4.3. Recall that in Section 4.3, we assume that we have access to a polynomial-time algorithm \(A\) that gives a \(\beta\)-approximate solution for Problem (SUB-DUAL(z,K)). We will use algorithm \(A\) within the Ellipsoid method, as part of our approximate separation oracle.

In Step 1 of the fair Ellipsoid-based algorithm, we apply the Ellipsoid method, outlined in Algorithm 3, to solving Problem (FAIR-DUAL). Here, we restate Problem (FAIR-DUAL):

\[
\text{FAIR-DUAL} = \min_{\rho \geq 0, z \geq 0} \rho + \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z_{ij} \right)
\]

\[
\text{s.t.} \quad \sum_{i \in S} \left( \sum_{j=1, j \neq i}^{n} z_{ij} w_j - \sum_{j=1, j \neq i}^{n} z_{ji} w_j \right) + \rho \geq \text{MS}(S), \quad \forall S : |S| \leq K
\]
For simplicity of notation, we can rewrite Problem (fair-dual) in the form of \( \min \{ \mathbf{d}^\top \mathbf{s} : \mathbf{A}s \geq \mathbf{b}, s \geq 0 \} \).

Here, \( \mathbf{s} = (\mathbf{z}, \rho) \in \mathbb{R}^{n^2+1} \) is the vector of decision variables. \( \mathbf{d} \) is a vector of size \( n^2 + 1 \) chosen such that \( \mathbf{d}^\top \mathbf{s} = \rho + \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z_{ij} \right) \). \( \mathbf{A} \) is a \( N \times (n^2 + 1) \) matrix and \( \mathbf{b} \) is a vector of size \( N \), where \( N = \{ S \subseteq [n] : |S| \leq K \} \). Let each row of \( \mathbf{A} \) and \( \mathbf{b} \) be indexed by a set \( S \) with \( |S| \leq K \). The matrix \( \mathbf{A} \) is chosen such that \( \mathbf{a}_S = \sum_{i \in S} \left( \sum_{j \in [n], j \neq i} z_{ij}w_i - \sum_{j \in [n], j \neq i} z_{ij}w_j \right) + \rho \), where \( \mathbf{a}_S \) is the row vector in \( \mathbf{A} \) indexed by set \( S \). The vector \( \mathbf{b} \) is chosen such that \( \mathbf{b}_S = \mathbf{m}_S(S) \), where \( \mathbf{b}_S \) is component of \( \mathbf{b} \) indexed by set \( S \).

Within the Ellipsoid method, we keep track of the following quantities: (i) \( (\mathbf{z}, \rho) \in \mathbb{R}^{n^2+1} \), the center of the current ellipsoid, which is also the current solution to the dual problem; note that this solution might not be feasible for the dual problem. (ii) \( (\mathbf{z}^*, \rho^*) \), the best feasible solution to the dual problem we have found so far. We initialize \( (\mathbf{z}^*, \rho^*) \) to be \( (0^{n^2}, 1) \), where \( 0^{n^2} \) is a zero vector of length \( n^2 \); this is always a feasible solution to Problem (fair-dual). (iii) the current best objective \( \text{OBJ} \). We initialize it to be 1, which is the objective of \( (0^{n^2}, 1) \). (iv) a positive-definite matrix \( \mathbf{D} \in \mathbb{R}^{(n^2+1) \times (n^2+1)} \), which represents the shape of the ellipsoid. (v) a collection \( \mathcal{V} \) of sets that have violated the dual fairness constraint during the execution of the Ellipsoid method.

At a high level, the Ellipsoid method for Problem (fair-dual) (Algorithm 3) works in the following way. At each iteration, it generates an ellipsoid \( E \), which is centered at the current solution \( \mathbf{s} = (\mathbf{z}, \rho) \), and defined as:

\[
E = \{ \mathbf{x} : (\mathbf{x} - \mathbf{s})^\top \mathbf{D}^{-1}(\mathbf{x} - \mathbf{s}) \leq 1 \}.
\]

By design of the Ellipsoid method, the ellipsoid \( E \) always contains the intersection of the feasibility region of Problem (fair-dual) and the half space \( \{ \mathbf{s} : \mathbf{d}^\top \mathbf{s} < \text{OBJ} \} \).

In Step 2(a), the algorithm first attempts to check if the current solution is feasible and improves our objective, via finding a violated constraint. There are three types of violated constraints that we consider: (i) the objective constraint is violated if the current solution \( (\mathbf{z}, \rho) \) does not yield an objective less than \( \text{OBJ} \); (ii) the non-negativity constraint is violated if any of the entries in \( (\mathbf{z}, \rho) \) is negative; (iii) the dual fairness constraint is violated if we can find a set \( S \) with \( |S| \leq K \) such that \( \mathbf{m}_S(S, \mathbf{z}) > \rho \). Note that when examining the dual fairness constraint, instead of examining the dual fairness constraint for every \( S \) such that \( |S| \leq K \), we instead use an approximate separation oracle that relies on algorithm \( \mathcal{A} \). First, it applies the \( \beta \)-approximate algorithm \( \mathcal{A} \) to Problem (sub-dual \((\mathbf{z}, K)\)) and gets \( S_{\mathcal{A}} \) with \( |S_{\mathcal{A}}| \leq K \) such that \( \mathbf{m}_S(S_{\mathcal{A}}, \mathbf{z}) \geq \beta \cdot \mathbf{m}_S(\mathbf{z}, K) \). Then, if \( \mathbf{m}_S(S_{\mathcal{A}}, \mathbf{z}) > \rho \), set \( S_{\mathcal{A}} \) has violated the dual fairness constraint. Otherwise, the solution \( (\mathbf{z}, \rho) \) is declared feasible. In the proof of Theorem 3, we will show that the use of the approximate separation oracle might cause the solution \( (\mathbf{z}^*, \rho^*) \) returned by the Ellipsoid method to be infeasible for Problem (fair-dual); however, the objective \( \text{OBJ} \) returned by the Ellipsoid method would stay close to the true optimal objective fair-dual.

If we have found one of the constraints violated in Step 2(a), which can be written as \( \mathbf{a}^\top \mathbf{s} < \mathbf{b} \) for some \( \mathbf{a} \in \mathbb{R}^{n^2+1} \) and \( \mathbf{b} \in \mathbb{R} \), we then go to Step 2(b) and construct a new ellipsoid \( E' \). The volume of the new ellipsoid \( E' \) is only a fraction of the volume of the previous ellipsoid \( E \). In addition, this new ellipsoid has a new center \( \mathbf{s}' = (\mathbf{z}', \rho') \) that satisfies \( \mathbf{a}^\top \mathbf{s'} > \mathbf{a}^\top \mathbf{s} \). We then start a new iteration, in which we check whether the new solution \( (\mathbf{z}', \rho') \) is feasible and improves our objective.
Algorithm 3 The Ellipsoid method for Problem (fair-dual)

Input: Starting solution \((z_0, \rho_0)\), starting matrix \(D_0\), maximum number of iterations \(t_{\text{max}}\). Here, we rewrite Problem (fair-dual) in the form of \(\min \{d^T s : As \geq b, s \geq 0\}\). The matrix \(A\) is chosen such that \(a_S^T s = \sum_{i \in S} (\sum_{j = 1, j \neq i} w_j - \sum_{j = 1, j \neq i} z_{ij} w_j) + \rho\), where \(a_S\) is the row vector in \(A\) indexed by set \(S\). The vector \(b\) is chosen such that \(b_S = \text{ms}(S)\), where \(b_S\) is component of \(b\) indexed by set \(S\).

Output: (i) A collection \(V\) of sets that have violated constraints. (ii) An optimal, feasible solution \((z^*, \rho^*) \geq 0\). (iii) Optimal objective \(\text{OBJ}\).

1. Initialization. \((z, \rho) = (z_0, \rho_0), (z^*, \rho^*) = (0, 1)\), \(\text{OBJ} = 1\), \(D = D_0\), \(V = \emptyset\), and \(t = 0\).

2. While \(t \leq t_{\text{max}}\):
   (a) Find a violated constraint.
      - Check if we can reduce the objective further. If \(\rho + \sum_{i = 1}^n \delta \cdot w_i \cdot \left(\sum_{j = [n], j \neq i} z_{ij}\right) \geq \text{OBJ}\), set \(a = -d\) and go to Step 2(b).
      - Check if the non-negativity constraints hold. If \(\rho < 0\), set \(a = e_{n+1}\); else if \(z_{ij} < 0\) for some \((i, j)\), set \(a = e_{n+j}\), where \(e_k\) denote the vector with a 1 in the \(k\)th coordinate and 0 elsewhere. Go to Step 2(b).
      - Check if the dual fairness constraints holds using the approximate separation oracle.
        - Apply the \(\beta\)-approximate algorithm \(A\) to Problem (sub-dual(\(z, K\))) that returns \(S_A\) with \(|S_A| \leq K\) such that \(\text{ms-cost}(S_A, z) \geq \beta \cdot \text{sub-dual}(z, K)\).
        - If \(\text{ms-cost}(S_A, z) > \rho\), then set \(S_A\) is violating the constraint. Set \(a = a_S\) and add \(S_A\) to \(V\).
      - If we have found no violated constraint, update our best feasible solution and its objective:
        \[(z^*, \rho^*) \leftarrow (z, \rho)\] \hspace{1cm} \(\text{OBJ} \leftarrow \rho^* + \sum_{i = 1}^n \delta \cdot w_i \cdot \left(\sum_{j = [n], j \neq i} z_{ij}^*\right)\).
        Then, go back to the start of Step 2 and re-enter the while loop.
   (b) Use the violated constraint to decrease the volume of the ellipsoid and find a new solution.
      \[\frac{\text{Da}}{\sqrt{\text{Da}^T \text{Da}}} \cdot \frac{1}{n^2 + 2} \cdot \frac{\text{Da}}{\sqrt{\text{Da}^T \text{Da}}} = \frac{(n^2 + 1)^2}{(n^2 + 1)^2 - 1} \cdot \left(\text{Da}^T \text{Da} - \frac{2}{n^2 + 2} \text{Da}^T \text{Da} - \text{Da}^T \text{Da}\right)\]
      \[\text{D} \leftarrow \frac{(n^2 + 1)^2}{(n^2 + 1)^2 - 1} \left(\text{Da}^T \text{Da} - \frac{2}{n^2 + 2} \text{Da}^T \text{Da} - \text{Da}^T \text{Da}\right)\]
      \[\text{t} \leftarrow t + 1\]
   (c) \(t \leftarrow t + 1\).
3. The ellipsoid is sufficiently small. Return \(V, (z^*, \rho^*), \text{OBJ}\).

On the other hand, if we have found no violated constraint, this means that our current solution \((z, \rho)\) is (approximately) feasible and also improves the objective \(\text{OBJ}\) obtained by the previous best feasible solution \((z^*, \rho^*)\). If this happens, we then set the current solution to be our current best feasible solution, i.e., \((z^*, \rho^*) \leftarrow (z, \rho)\), and update the current optimal objective \(\text{OBJ}\) to be the objective obtained by the current solution. Then, we start a new iteration, and seeks to find a feasible solution that can further reduce our objective function.

The Ellipsoid method for Problem (fair-dual) terminates when the ellipsoid \(E\) is sufficiently small. Recall that the ellipsoid \(E\) always contains the intersection of the feasibility region of Problem (fair-dual) and the half space \(\{ s : d^T s < \text{OBJ}\}\). Intuitively, when the ellipsoid gets reduced to a sufficiently small volume, it is unlikely that there exists a feasible solution that can further reduce our objective. It is shown, in Bertsimas
and Tsitsiklis (1997), that given a separation oracle, the Ellipsoid method is guaranteed to terminate in $t_{\text{max}} = O(n^{12}\log(nw_{\text{max}}))$ iterations\footnote{In our implementation of the Ellipsoid method, we tune $t_{\text{max}}$ for better performance in runtime.}. When the Ellipsoid method terminates, it returns the following: (i) the collection $\mathcal{V}$ of sets that have violated the dual fairness constraint; (ii) the optimal solution $(\mathbf{z}^*, \rho^*)$ and (iii) the optimal objective OBJ. Note that this collection $\mathcal{V}$ would then play an important role in Step 2 of the fair Ellipsoid-based algorithm. As discussed in Section 4.3, we then use the collection $\mathcal{V}$ to reduce the number of variables to be considered in the primal problem (FAIR) to a polynomial size.

Appendix C: Proof of Theorem 2

We will prove the NP-completeness of Problem $(\text{SUB-DUAL}(\mathbf{z}, K))$ by considering an arbitrary instance of the partition problem, and solving it using polynomially many instances of Problem $(\text{SUB-DUAL}(\mathbf{z}, K))$. Recall that the partition problem is stated as follows: given a set of positive integers $\{a_1, a_2, \ldots, a_n\}$, we would like to determine if there exists a subset $S \subset [n]$ such that $\sum_{i \in S} a_i = \frac{1}{2} \sum_{i \in [n]} a_i \triangleq L$, and find $S$ if it exists. The partition problem is known to be NP-complete Hayes (2002).

Before proceeding, let us first define an auxiliary optimization problem, for some given vector $\mathbf{c}$:

\[
\text{SUB-DUAL-AUX}(\mathbf{c}, K) = \max_{S: |S| \leq K} \frac{\sum_{i \in S} w_i}{1 + \sum_{i \in S} w_i} - \sum_{i \in S} c_i'.
\]

We denote the objective function of Problem $(\text{SUB-DUAL-AUX}(\mathbf{c}', K))$ as

\[
\text{MS-COST-AUX}(S, \mathbf{c}') := \frac{\sum_{i \in S} w_i}{1 + \sum_{i \in S} w_i} - \sum_{i \in S} c_i'.
\]

Recall that Problem $(\text{SUB-DUAL}(\mathbf{z}, K))$ is defined as:

\[
\text{SUB-DUAL}(\mathbf{z}, K) = \max_{S: |S| \leq K} \frac{\sum_{i \in S} w_i}{1 + \sum_{i \in S} w_i} - \sum_{i \in S} c_i(\mathbf{z}) = \max_{S: |S| \leq K} \text{MS-COST}(S, \mathbf{z}),
\]

where $c_i(\mathbf{z})$ is defined in (5). Note that Problem $(\text{SUB-DUAL-AUX}(\mathbf{c}', K))$ only differs from Problem $(\text{SUB-DUAL}(\mathbf{z}, K))$ in the second term of the objective function, where $\mathbf{c}'$ can be any vector and does not depend on $\mathbf{z}$. Our proof of Theorem 2 consists of two parts. In Part 1, we reduce an arbitrary instance of the partition problem to a particular instance of Problem $(\text{SUB-DUAL-AUX}(\mathbf{c}', K))$. In Part 2, we show that this instance of Problem $(\text{SUB-DUAL-AUX}(\mathbf{c}', K))$ can be further reduced to a polynomial number of instances of Problem $(\text{SUB-DUAL}(\mathbf{z}, K))$.

Part 1. Let $\{a_1, a_2, \ldots, a_n\}$ be an instance of the partition problem. We first reduce it to the auxiliary optimization problem by creating an instance of Problem $(\text{SUB-DUAL-AUX}(\mathbf{c}', K))$ as follows. For each $i \in [n]$, let $w_i = a_i/L$ and $c_i' = a_i/4L$. Let $K = n/2$. Additionally, let $A(S) := \sum_{i \in S} a_i$. We can rewrite the objective function of Problem $(\text{SUB-DUAL-AUX}(\mathbf{c}', K))$ to be a function dependent on $A(S)$:

\[
\frac{\sum_{i \in S} w_i}{1 + \sum_{i \in S} w_i} - \sum_{i \in S} c_i' = \frac{A(S)}{L + A(S)} - \frac{A(S)}{4L} = \frac{3LA(S) - (A(S))^2}{4L^2 + 4LA(S)} := f(A(S)),
\]

where we let $f(x) = \frac{3Lx - x^2}{4L^2 + 4Lx}$ for $x \geq 0$. Differentiating $f(x)$ with respect to $x$ gives

\[
f'(x) = \frac{-x^2 - 2Lx + 3L^2}{4L(L + x)^2}.
\]
Taking \( f'(x) = 0 \) gives \( x = L \). Since \( f'(x) < 0 \) for \( x > L \) and \( f'(x) > 0 \) for \( 0 \leq x < L \), we have that for \( x \geq 0 \), \( f(x) \) is uniquely maximized at \( x = L \) and \( \max_{x \geq 0} f(x) = f(L) = \frac{1}{4} \).

We now show that we can determine whether a partition \( S \) exists and, if it exists, find \( S \) if and only if we can find the optimal set \( S^* \) for instance of Problem \((\text{sub-dual-aux}(c',K))\) and it attains optimal objective value of \( 1/4 \). Suppose a partition \( S^* \) exists\(^\text{18}\) such that \( A(S^*) = \sum_{i \in S^*} a_i = L \), we have

\[
\frac{1}{4} \geq \text{sub-dual-aux}(c',K) \geq f(A(S)) = f(A(S^*)) = f(L) = \frac{1}{4}.
\]

Hence, the inequalities above should be equalities. We have that \( S^* \) is the optimal set for Problem \((\text{sub-dual-aux}(c',K))\) and the optimal objective value is \( f(A(S^*)) = 1/4 \). Reversely, if we find \( S^* = \arg \max_{S \subseteq [n]} |S| \leq K \text{sub-dual-aux}(c',K) \) and \( \max_{S \subseteq [n]} f(A(S)) = f(L) = \frac{1}{4} \), we must have that \( A(S^*) = \sum_{i \in S^*} a_i = L \), and \( S^* \) is thus a solution to the partition problem. The discussion above shows that we can reduce the partition problem to the instance of Problem \((\text{sub-dual-aux}(c',K))\). So, Problem \((\text{sub-dual-aux}(c',K))\) is NP-complete.

**Part 2.** Now, it suffices to show that the instance of Problem \((\text{sub-dual-aux}(c',K))\) defined above can be solved using polynomially many instances of Problem \((\text{sub-dual}(z,K))\). Without loss of generality, assume that \( a_1 \geq a_2 \geq \cdots \geq a_n \). Before proceeding, let us first make the following definitions.

1. For every \( k \in [n] \), let \((\text{sub-dual-aux}(k))\) denote an instance of Problem \((\text{sub-dual-aux}(c',K))\) with \( k \) items, where \( w = (w_1, \ldots, w_k) = (a_1/L, \ldots, a_k/L) \), \( c' = (c'_1, \ldots, c'_k) = (a_1/L, \ldots, a_k/L) \) and \( K = n/2 \). The instance of Problem \((\text{sub-dual-aux}(c',K))\) that we considered in Part 1 of the proof is \((\text{sub-dual-aux}(n))\).

2. For every \( k \in [n] \), let \((\text{sub-dual}(k))\) denote an instance of Problem \((\text{sub-dual}(z,K))\) with \( k \) items, where \( w = (w_1, \ldots, w_k) = (a_1/L, \ldots, a_k/L) \), and \( K = n/2 \). We choose \( z^{(k)} \) such that \( c_i(z^{(k)}) = c'_i = a_i/4L \) for \( i \in [k-1] \). Note that such \( z^{(k)} \) always exists, since the linear system

\[
\sum_{j \neq i} (z^{(k)}_j - z^{(k)}_i) w_j = \frac{a_i}{4L}, \quad \forall i \in [k-1]
\]

can be written as

\[
\sum_{j \neq i} \mathbb{1}_{\{1 < j\}} \frac{a_j}{L} x_{ij} = \frac{a_i}{4L}, \quad \forall i \in [k-1]
\]

where \( x_{ij} = z^{(k)}_j - z^{(k)}_i \) for all \( i < j \). The rows of this \((k-1) \times (k-1)(k-2)\) linear system are linearly independent, and hence there exists a feasible solution. However, via row operations, one can check that the values of \( c_1(z^{(k)}), \ldots, c_{k-1}(z^{(k)}) \) uniquely determine the value of \( c_k(z^{(k)}) \) to be

\[
c_k(z^{(k)}) = -\frac{\sum_{i < k} a_i^2}{4La_k}.
\]

Note that the only difference between the instance \((\text{sub-dual}(k))\) and \((\text{sub-dual-aux}(k))\) is in the difference between \( c_k(z) \) and \( c'_k \). Additionally, we have \( c_k(z^{(k)}) = -(\sum_{i < k} a_i^2)/(4La_k) \leq a_k/4L = c'_k \) because \( a_i \)'s are all positive integers.

\(^\text{18}\)Note that we can assume without loss of generality that \(|S^*| \leq n/2 \). This is because if \( S^* \) satisfies \( \sum_{i \in S} a_i = \frac{1}{2} \sum_{i \in [n]} a_i \), we must also have \( \sum_{i \in [n] \setminus S^*} a_i = \frac{1}{2} \sum_{i \in [n]} a_i \). That is, \( S^* \) and \([n] \setminus S^* \) are both valid partitions.
Having defined the two types of instances, our proof of Part 2 proceeds in three steps: (i) first, we establish an important relationship between the two functions \(\text{MS-COST} \) and \(\text{MS-COST-AUX}\); (ii) next, we show that for any \(k \in [n]\), we can solve the instance \((\text{SUB-DUAL-AUX}^{(k)})\) if we can solve \((\text{SUB-DUAL}^{(k)})\) and \((\text{SUB-DUAL-AUX}^{(k-1)})\); (iii) based on that, we use a recursive approach to show that the instance \((\text{SUB-DUAL-AUX}^{(n)})\) considered in Part 1 of the proof can be solved as long as we can solve \((\text{SUB-DUAL}^{(n)}), \ldots, (\text{SUB-DUAL}^{(2)})\), which are \(n - 1\) instances of Problem \((\text{SUB-DUAL}(z, K))\).

**Step 1.** Fix \(k \in [n]\). Let us start by considering the two instances \((\text{SUB-DUAL}^{(k)})\) and \((\text{SUB-DUAL-AUX}^{(k)})\). Recall that the objective function of \((\text{SUB-DUAL}^{(k)})\) is:

\[
\text{MS-COST}(S, z^{(k)}) = \frac{\sum_{i \in S} w_i}{1 + \sum_{i \in S} w_i} - \sum_{i \in S} c_i(z^{(k)})
\]

and the objective function of \((\text{SUB-DUAL-AUX}^{(k)})\) is

\[
\text{MS-COST-AUX}(S, c) = \frac{\sum_{i \in S} w_i}{1 + \sum_{i \in S} w_i} - \sum_{i \in S} c_i'.
\]

Since \(c_i(z^{(k)}) = c'_i\) for all \(i \in [k - 1]\), we must have the following:

- For any set \(S\) such that \(k \notin S\), we must have \(\text{MS-COST}(S, z^{(k)}) = \text{MS-COST-AUX}(S, c')\).
- For any set \(S\) such that \(k \in S\), we must have

\[
\text{MS-COST}(S, z^{(k)}) = \text{MS-COST-AUX}(S, c') + \frac{a_k}{4L} + \frac{\sum_{i < k} a_i^2}{4L a_k} \geq \text{MS-COST-AUX}(S, c')
\]

**Step 2.** Now, suppose that we have solved \((\text{SUB-DUAL}^{(k)})\) with \(S_k^*\) as its optimal solution, and we have also solved \((\text{SUB-DUAL-AUX}^{(k-1)})\) with \(S'_{k-1}\) as its optimal solution. We claim that the optimal solution to \((\text{SUB-DUAL-AUX}^{(k)})\), \(S'_k\), would then be the best set among \(S_k^*\) and \(S'_{k-1}\).

To see why, we divide our arguments into the following cases:

- **Case 1:** \(k \notin S_k^*\). In this case, we claim that \(S_k^*\) is an optimal solution to \((\text{SUB-DUAL-AUX}^{(k)})\). Let \(S' \in [k], |S'| \leq K\) be any set that is different from \(S_k^*\). There are again two possibilities:
  - If \(k \notin S'\), we have

\[
\text{MS-COST-AUX}(S', c') = \text{MS-COST}(S', z^{(k)}) \leq \text{MS-COST}(S_k^*, z^{(k)}) = \text{MS-COST-AUX}(S_k^*, c')
\]

  where the two equalities follow from Step 1.
  - If \(k \in S'\), we have

\[
\text{MS-COST-AUX}(S', c') \leq \text{MS-COST}(S', z^{(k)}) \leq \text{MS-COST}(S_k^*, z^{(k)}) = \text{MS-COST-AUX}(S_k^*, c')
\]

  where the first inequality and the last equality follow from Step 1.

Since we have \(\text{MS-COST-AUX}(S'^{*}, c) \leq \text{MS-COST-AUX}(S_k^*, c)\) for any set \(S'^{*}\), \(S_k^*\) must be an optimal solution to \((\text{SUB-DUAL-AUX}^{(k)})\).

- **Case 2:** \(k \in S_k^*\). In this case, we claim that the optimal solution to \((\text{SUB-DUAL-AUX}^{(k)})\), which we denote by \(S'_k\), is either \(S_k^*\) or \(S'_{k-1}\). Let us again consider the two possibilities:
If \( k \in S'_k \), we must have

\[
\text{MS-COST-AUX}(S'_k, c') = \text{MS-COST}(S'_k, z^{(k)}) - \left( \frac{a_k}{4L} + \frac{\sum_{i<k} a_i^2}{4La_k} \right)
\leq \text{MS-COST}(S'_k, z^{(k)}) - \left( \frac{a_k}{4L} + \frac{\sum_{i<k} a_i^2}{4La_k} \right)
= \text{MS-COST-AUX}(S'_k, c').
\]

where the equalities follow from Step 1. Hence, in this case, \( S'_k \) is an optimal solution to (SUB-DUAL-AUX\((k)\)).

• If \( k \notin S'_k \). In this case, an optimal solution to (SUB-DUAL-AUX\((k)\)) is simply the optimal solution to (SUB-DUAL-AUX\((k-1)\)); that is, \( S'_{k-1} \).

To summarize, our case discussion above shows that to solve (SUB-DUAL-AUX\((k)\)), it suffices to solve \((\text{SUB-DUAL}^{(k)})\) and \((\text{SUB-DUAL-AUX}^{(k-1)})\).

\textbf{Step 3.} Finally, we consider the instance (SUB-DUAL-AUX\((n)\)) from Part 1 of the proof. Since we have shown in Step 2 that for any \( k \in [n] \), we can solve (SUB-DUAL-AUX\((k)\)) as long as we can solve (SUB-DUAL\((k)\)) and (SUB-DUAL-AUX\((k-1)\)), we can solve (SUB-DUAL-AUX\((n)\)) in a recursive manner. This requires us to solve at most \( n-1 \) instances of Problem (SUB-DUAL\((z, K)\)), i.e., (SUB-DUAL\((n)\)), (SUB-DUAL\((n-1)\)), \ldots, (SUB-DUAL\((2)\)) and one instance of Problem (SUB-DUAL-AUX\((c', K)\)), i.e., (SUB-DUAL-AUX\((1)\)), for which the solution is trivial to obtain.

In summary, in Part 2 we have shown that (SUB-DUAL-AUX\((n)\)) can be solved using a polynomial number of instances of Problem (SUB-DUAL\((z, K)\)). Together, Part 1 and 2 of the proof show that an arbitrary instance of the partition problem can be solved using polynomially many instances of Problem (SUB-DUAL\((z, K)\)). This establishes the NP-completeness of Problem (SUB-DUAL\((z, K)\)). \( \blacksquare \)

\textbf{Appendix D: Proof of Theorem 3}

Our proof of Theorem 3 consists of two segments. In Segment 1, we show that our fair Ellipsoid-based algorithm gives a \( \beta \)-approximate solution for Problem (FAIR), and the returned solution requires us to randomize over a polynomial number of sets. In Segment 2, we show that the running time of our fair Ellipsoid-based algorithm is polynomial in the size of our input.

\textbf{Segment 1.} Before proceeding, recall that in Section 4.3, we define \( \mathcal{V} = \{ S : |S| \leq K \} \) as the collection of sets that violate the dual fairness constraint. The first segment of our proof consists of three parts: (i) We first show that if we apply the Ellipsoid method equipped with a \( \beta \)-approximate separation oracle to the dual problem (FAIR-DUAL), we will end up with a dual objective \( r' \) that is not too far away from the optimal primal objective FAIR. That is, it satisfies \( r' \geq \beta \cdot \text{FAIR} \). (ii) We then construct a auxiliary dual problem (FAIR-DUAL-AUX), in which we only keep the dual fairness constraints for \( S \in \mathcal{V} \); and the corresponding auxiliary primal problem (FAIR-AUX), in which we enforce \( p(S) = 0 \) for all \( S \notin \mathcal{V} \). The auxiliary primal problem is the one we solve in Step 2 of our algorithm. We will show that the optimal objectives of both Problem (FAIR-DUAL-AUX) and (FAIR-AUX) exceeds \( r' \), and hence exceeds \( \beta \cdot \text{FAIR} \). (iii) Finally, we show that the number of sets in \( \mathcal{V} \) is polynomial, which implies that \( |\{ S : \hat{p}(S) > 0 \}| \) is polynomial.

\textbf{Part 1.} As described in Step 1 of the fair Ellipsoid-based algorithm, suppose that we apply the Ellipsoid method to the dual problem (FAIR-DUAL), and use algorithm \( \mathcal{A} \) as the \( \beta \)-approximate separation oracle. Let
(z’, ρ’) be the solution we get at the end of the Ellipsoid method, and let 
\[ r' = \rho' + \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z'_{ij} \right) \]
denote its dual objective. We first note that the solution (z’, ρ’) is not necessarily feasible for Problem (FAIR-DUAL). This is because when we examine the feasibility of this solution, our separation oracle first solves Problem (SUB-DUAL(z’, K)), and obtained a set \( S_A \) such that \( \text{ms-cost}(S_A, z') \geq \beta \cdot \text{SUB-DUAL}(z', K) \).

The Ellipsoid method has declared solution (z’, ρ’) feasible because \( \text{ms-cost}(S_A, z') \leq \rho' \). However, it is still possible that \( \text{SUB-DUAL}(z', K) = \max_{|S| \leq K} \text{ms-cost}(S, z') > \rho' \).

Suppose that we solve the dual problem (FAIR-DUAL) by additionally fixing \( z = z' \). This problem has a trivial solution because \( \rho \) is the only decision variable, and it must be chosen such that the inequality in Equation (3) is tight. That is, we set \( \rho = \text{SUB-DUAL}(z', K) \) and the optimal objective is

\[ r'_\rho := \text{SUB-DUAL}(z', K) + \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z'_{ij} \right) \]

Clearly, we must have FAIR-DUAL \( \leq r'_\rho \) since fixing \( z = z' \) is equivalent to introducing more constraints into Problem (FAIR-DUAL). Note that we also have

\[ r' = \rho' + \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z'_{ij} \right) \geq \beta \cdot \text{SUB-DUAL}(z', K) + \beta \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z'_{ij} \right) = \beta \cdot r'_\rho, \]

where the inequality follows from \( \rho' \geq \text{ms-cost}(S_A, z') \geq \beta \cdot \text{SUB-DUAL}(z', K) \) and \( z'_{ij} \geq 0 \). Together, the two inequalities above give

\[ r' \geq \beta \cdot r'_\rho \geq \beta \cdot \text{FAIR-DUAL} = \beta \cdot \text{FAIR}, \tag{13} \]

where the last equality follows from strong duality\(^{19}\). The inequality in Equation (13) is a key inequality that we will continue to work with in Part 2.

**Part 2.** In Part 2, we first create an auxiliary version of the dual problem, defined as follows:

\[
\begin{align*}
\text{FAIR-DUAL-AUX} &= \min_{\rho \geq 0, x \geq 0} \rho + \sum_{i=1}^{n} \delta \cdot w_i \cdot \left( \sum_{j \in [n], j \neq i} z_{ij} \right) \\
\text{s.t.} \sum_{i \in S} \left( \sum_{j=1, j \neq i}^{n} z_{ij} w_j - \sum_{j=1, j \neq i}^{n} z_{ji} w_j \right) + \rho &\geq \text{ms}(S), \quad \forall S \in \mathcal{V}
\end{align*}
\]

Note that the auxiliary dual problem is different from Problem (FAIR-DUAL), because in the auxiliary dual problem, we only enforce the constraints on sets \( S \in \mathcal{V} \), where \( \mathcal{V} \) is the collection sets for which the dual fairness constraint has been violated when we solve Problem (FAIR-DUAL). Observe that if we apply the Ellipsoid method with the \( \beta \)-approximate separation oracle to solve Problem (FAIR-DUAL-AUX), the solution we obtain would still be (z’, ρ’), which gives objective value \( r' \). This is because the unviolated constraints do not impact any of the iteration in the Ellipsoid method.

Similar to our arguments in Part 1, we note that when we apply the Ellipsoid method using our approximate separation oracle, we essentially increases the feasibility region of the linear program. Hence, the solution (z’, ρ’) we found at the end of the Ellipsoid method might not be feasible and the objective we obtain is less

\(^{19}\) Since both Problem (FAIR) and Problem (FAIR-DUAL) are feasible and bounded, they both admit an optimal solution. This allows us to invoke strong duality (see Theorem 4.4 in Bertsimas and Tsitsiklis (1997)), which suggests that FAIR = FAIR-DUAL.
than or equal to the actual optimal objective. That is, we must have \( r' \leq \text{FAIR-DUAL-AUX} \). This, along with the inequality we established in Equation (13), gives

\[
\text{FAIR-DUAL-AUX} \geq r' \geq \beta \cdot \text{FAIR}.
\]

Consider the primal counterpart to the auxiliary dual problem \((\text{FAIR-DUAL-AUX})\) defined below:

\[
\text{FAIR-AUX} = \max_{p(S) \geq 0, S \in V} \sum_{S \subseteq V} p(S) \cdot \text{MS}(S)
\]

s.t. \[
\frac{w_j}{w_i} \sum_{S : i \in S} p(S) - \sum_{S : j \in S} p(S) \leq \delta \qquad i, j \in [n]
\]

\[
\sum_{S : |S| \in V} p(S) \leq 1
\]

\[\text{(FAIR-AUX)}\]

Essentially, this differs from Problem \((\text{FAIR})\) in that here, we set \( p(S) = 0 \) for all \( S \notin V \). This is the primal problem that we solve in Step 2 of the fair Ellipsoid-based algorithm. Let \( \hat{p}(S) \) denote the optimal solution to Problem \((\text{FAIR-AUX})\). By the strong duality\(^{20}\), we have \( \text{FAIR-AUX} = \text{FAIR-DUAL-AUX} \geq \beta \cdot \text{FAIR} \). We have thereby showed that our algorithm returns a solution \( \hat{p}(S) \) that is an \( \beta \)-approximate feasible solution to Problem \((\text{FAIR})\).

**Part 3.** Since we have \( \hat{p}(S) = 0 \) for all \( S \notin V \), the number of sets that we need to randomize over, i.e. \(|\{S : \hat{p}(S) > 0\}|\), is bounded by \(|V|\). Recall that in the dual problem \((\text{FAIR-DUAL})\), we have \( O(n^2) \) variables. By Bertsimas and Tsitsiklis (1997), given a separation oracle, the Ellipsoid method would solve \((\text{FAIR-DUAL})\) in at most \( O(n^{12} \log(n \log w_{\text{max}})) \) iterations. Within each iteration, one violated constraint is identified. Hence, we must have that \(|V|\), i.e., the number of sets for which the dual fairness constraints in \((\text{FAIR-DUAL})\) is violated, is polynomial in \( n \) and \( \log(w_{\text{max}}) \). This thus proves that \(|\{S : \hat{p}(S) > 0\}|\) is polynomial in the size of our input.

**Segment 2.** In the second segment of the proof, we remark on the running time of our fair Ellipsoid-based algorithm. In Step 1 of the algorithm, the Ellipsoid method equipped with a polynomial-time separating oracle runs in polynomial time. In Step 2 of the algorithm, solving a linear program with polynomial number of variables also takes polynomial time. Hence, given a polynomial-time separation oracle, the total time taken by the fair Ellipsoid-based algorithm is polynomial in the size of our input. □

**Appendix E: Proof of Theorem 4**

Let \( S^* \in \arg \max_{S : |S| \leq K} \text{MS-COST}(S, z) \) and let \( W^* = \sum_{i \in S^*} w_i \). We must have

\[
\text{SUB-DUAL}(z, K) = \text{MS-COST}(S^*, z) \leq \text{KP}(W^*, K) \leq \max_{W \geq 0} \text{KP}(W, K) .
\]

The first inequality follows from the fact that (i) \( S^* \) is a feasible solution to Problem \( \text{KP}(W^*, K) \) since \( |S^*| \leq K \) and \( \sum_{i \in S^*} w_i = W^* \) by definition, and (ii)

\[
\text{MS-COST}(S^*, z) = \frac{\sum_{i \in S^*} w_i}{1 + \sum_{i \in S^*} w_i} - \sum_{i \in S^*} c_i = \sum_{i \in S^*} \left( \frac{w_i}{1 + W^*} - c_i \right),
\]

\(^{20}\) Here, both Problem \((\text{FAIR-AUX})\) and Problem \((\text{FAIR-DUAL-AUX})\) are again feasible and bounded, they both admit an optimal solution. For Problem \((\text{FAIR-AUX})\), \( p(S) = 0 \) for all \( S \notin V \) is a feasible solution; its objective is upper bound by 1. For Problem \((\text{FAIR-DUAL-AUX})\), \( p = 1, z = 0 \) is a feasible solution; its objective is lower bounded by 0. This again allows us to invoke strong duality (see Theorem 4.4 in Bertsimas and Tsitsiklis (1997)), which suggests that \( \text{FAIR-AUX} = \text{FAIR-DUAL-AUX} \).
where the right-hand side (i.e., $\sum_{i \in S^*} \left( \frac{w_i}{1 + W^*} - c_i \right)$) is the objective value of the knapsack problem ($\text{KP}(W^*, K)$) at set $S^*$, which is clearly upper bounded by its optimal objective value $\text{KP}(W^*, K)$.

Conversely, let $\max_{W \geq 0} \text{KP}(W, K)$ be obtained at capacity $\tilde{W}$ and with optimal set $\tilde{S}$, i.e., $\tilde{S}$ is the optimal solution to Problem $\text{KP}(\tilde{W}, K)$. We then have

$$\max_{W \geq 0} \text{KP}(W, K) = \text{KP}(\tilde{W}, K) = \sum_{i \in \tilde{S}} \left[ \frac{w_i}{1 + \tilde{W}} - c_i \right]$$

$$\leq \sum_{i \in \tilde{S}} \frac{w_i}{1 + \sum_{i \in \tilde{S}} w_i} - \sum_{i \in \tilde{S}} c_i = \text{MS-COST}(\tilde{S}, z) \leq \max_{|S| \leq K} \text{MS-COST}(S, z),$$

where the first inequality follows from the feasibility of $\tilde{S}$ for Problem $\text{KP}(\tilde{W}, K)$, which implies that $\sum_{i \in \tilde{S}} w_i \leq \tilde{W}$, and the second inequality follows from $|\tilde{S}| \leq K$.

Overall, Equations (14) and (15) together imply that $\text{SUB-DUAL}(z, K) = \max_{S:|S| \leq K} \text{MS-COST}(S, z) = \max_{W \geq 0} \text{KP}(W, K)$, as desired. ■

Appendix F: Proof of Theorem 5

The proof consists of two segments. In the first segment, we show that

$$\text{MS-COST}(S_{1/2}, z) \geq \frac{1}{2} \max_{W \in [0, \infty)} \text{KP}(W, K),$$

where $S_{1/2}$ is the assortment returned by Algorithm 1. In the second segment, we bound the running time of the algorithm.

Segment 1. We start by considering the LP relaxation of the original knapsack problem, i.e., Problem ($\text{KP-RELAX}(W, K)$), and noting that $\text{KP-RELAX}(W, K)$ is an upper bound of $\text{KP}(W, K)$. Therefore,

$$\max_{W \in [0, \infty)} \text{KP-RELAX}(W, K) \geq \max_{W \in [0, \infty)} \text{KP}(W, K).$$

(16)

Suppose that $\max_{W \in [0, \infty)} \text{KP-RELAX}(W, K)$ is obtained at capacity level $W^*$. That is,

$$\max_{W \in [0, \infty)} \text{KP-RELAX}(W, K) = \text{KP-RELAX}(W^*, K).$$

We now make the following claim:

Claim 1. In the collection of assortments $C$ maintained in Algorithm 1, there exists an assortment $S_{W^*}$ such that $S_{W^*}$ is an integer $1/2$ approximate feasible solution for Problem $\text{KP-RELAX}(W^*, K)$.

Given that Claim 1 holds, we then have

$$\text{MS-COST}(S_{W^*}, z) = \sum_{i \in S_{W^*}} \left( \frac{w_i}{1 + \sum_{i \in S_{W^*}} w_i} - c_i \right) \geq \sum_{i \in S_{W^*}} \left( \frac{w_i}{1 + W^*} - c_i \right) \geq \frac{1}{2} \text{KP-RELAX}(W^*, K),$$

(17)

where the first inequality follows from the feasibility of $S_{W^*}$ (i.e., $\sum_{i \in S_{W^*}} w_i \leq W^*$) and the second inequality follows from Claim 1. This then further gives

$$\text{MS-COST}(S_{1/2}, z) \geq \text{MS-COST}(S_{W^*}, z) \geq \frac{1}{2} \text{KP-RELAX}(W^*, K) \geq \frac{1}{2} \max_{W \in [0, \infty)} \text{KP}(W, K),$$

(18)

where the first inequality follows from $S_{1/2} = \arg \max_{S \in C} \text{MS-COST}(S, z)$ (Step 4 of Algorithm 1) and that $S_{W^*} \in C$; the second inequality follows from Equation (17); and the third inequality follows from Equation (16). This (i.e., Claim 1) thus proves the statement of Theorem 5.
In the following, we provide a proof for Claim 1. Our proof consists of five parts. In Part 1, we first show that $W^*$ is bounded by $W_{\text{max}}$, which is the maximum capacity level under which the capacity constraint is binding for the relaxed knapsack problem. This then allows us to justify the stopping rules in Step 2b and Step 3(b)iii of the algorithm; see Parts 2 and 3 of the proof. Recall that in Algorithm 1, we consider the two intervals $I_{\text{low}} = [0, W_{\text{th}}]$ and $I_{\text{high}} = [W_{\text{th}}, \infty)$ separately. In Part 2, we show that when we hit the stopping rule in Step 2b, we have reached the upper bound $W_{\text{max}}$ and there is no need to consider interval $I_{\text{high}}$. Similarly, in Part 3, we show that when we hit stopping rule in Step 3(b)iii, we have reached $W_{\text{max}}$ and there is no need to consider larger $W$'s. In Part 4 of the proof, we show that if $W^* \in I_{\text{low}}$, there exists a 1/2 approximate solution $S_{W^*} \in \mathcal{C}$. In Part 5, which is one of the most challenging parts of the proof, using induction, we show that if $W^* \in I_{\text{high}}$, there exists a 1/2 approximate solution $S_{W^*} \in \mathcal{C}$.

**Part 1: $W^*$ is bounded by $W_{\text{max}}$.** Let $W_{\text{max}}$ be the maximum capacity level $W$ for which the capacity constraint is binding for Problem \((\text{KP-relax}(W, K))\), i.e.,

$$W_{\text{max}} := \max \left\{ W \in [0, \infty) : \sum_{i \in [n]} w_i x_i^*(W) = W \right\} \text{ and } \sum_{i \in [n]} w_i x_i^*(W') < W' \quad \forall W' > W_{\text{max}}.$$  

(19)

Lemma 2 shows that $W_{\text{max}}$ is well-defined. The lemma states that if the capacity constraint is non-binding at some capacity level, Problem \((\text{KP-relax}(W, K))\) will have a non-binding capacity constraint for all higher capacity levels. Since the capacity constraint is binding at $W = 0$ and non-binding at $W = \sum_{i \in [n]} w_i$, there must exist $W_{\text{max}} \in [0, \sum_{i \in [n]} w_i]$ that satisfies the definition in Equation (19).

**Lemma 2.** Consider a capacity level $\bar{W} > 0$, and suppose that the capacity constraint is non-binding at the optimal basic solution $x^*(\bar{W})$ to \(\text{KP-relax}(\bar{W}, K)\). That is, $\sum_{i \in [n]} w_i x_i^*(\bar{W}) < \bar{W}$. Then, for any $W > \bar{W}$, we have $\sum_{i \in [n]} w_i x_i^*(W) < W$.

To show that $W^* \in \arg\max_{W \in [0, \infty]} \text{KP-relax}(W, K)$ is bounded by $W_{\text{max}}$, we use Lemma 3 below, which states that at capacity level $W^*$, the capacity constraint must be binding for the optimal solution of Problem \(\text{KP-relax}(W^*, K)\). Then, by definition of $W_{\text{max}}$ in Equation (19), we must have $W^* \leq W_{\text{max}} < \infty$.

**Lemma 3.** Let $W^* \in \arg\max_{W \in [0, \infty]} \text{KP-relax}(W, K)$ and define $x^* = x^*(W^*)$ as the optimal basic solution to Problem \((\text{KP-relax}(W^*, K))\). Then, the capacity constraint in Problem \((\text{KP-relax}(W^*, K))\) is binding. That is, $\sum_{i \in [n]} w_i x_i^* = W^*$.

**Part 2: hitting the stopping rule in Step 2b.** Here, we show that if we hit the stopping rule in Step 2b of Algorithm 1, we must have $W^* \in I_{\text{low}}$ and hence, we do not need to consider interval $I_{\text{high}}$, where $I_{\text{low}} = [0, W_{\text{th}}]$ and $I_{\text{high}} = [W_{\text{th}}, \infty)$. Recall that in Step 2b of Algorithm 1, we stop the algorithm when $u_{h_K}(W_{\text{th}}) < 0$, where $h_K$ is the index of the item with the $K$th highest utility-to-weight ratio. Note that if $u_{h_K}(W_{\text{th}}) < 0$, then all items with lower utility-to-weight ratios must also have negative utilities, i.e. $u_{h_j}(W_{\text{th}}) < 0$ for all $j \geq K$. Hence, the optimal solution $x^*(W_{\text{th}})$ for Problem \((\text{KP-relax}(W_{\text{th}}, K))\) should not include any of the items $h_j$ for all $j \geq K$. This implies that at $W_{\text{th}}$, the capacity constraint is no longer binding. (Recall that $W_{\text{th}} = w(H_K)$ is the sum of the weights of items $h_1, \ldots, h_K$.) Then, by Lemma 2, we have $W_{\text{max}} < W_{\text{th}}$. Since in Part 1 of the proof, we show that $W^* \in [0, W_{\text{max}}]$, then we must have $W^* \in I_{\text{low}}$. 

and we do not need to consider \( I_{\text{high}} \). This explains why we only go to Step 3 and evaluate \( I_{\text{high}} \) when \( u_{h,K}(W_{\text{th}}) \geq 0 \) in Algorithm 1.

**Part 3: hitting the stopping rule in Step 3b.** Here, we show that if we hit the stopping rule in Step 3b, we must have reached \( W_{\text{max}} \) and we do not need to consider larger \( W \)'s.

Before proceeding, we first make the following definitions. In Step 3b of Algorithm 1, we adaptively partition the interval \( I_{\text{high}} \). In particular, at each iteration, the algorithm updates two quantities: (i) the capacity change point \( W_{\text{next}} \); (ii) two sets of indices that represent the profile at \( W_{\text{next}} \): \( P_1 \) and \( P_0 \). We let \( W^{(k)}_{\text{next}} \) be the change point updated in the \( k \)th iteration of the while loop, and \( P^{(k)}_1, P^{(k)}_0 \) be the sets of indices updated in the \( k \)th iteration. Additionally, we let \( W^{(0)}_{\text{next}} = W_{\text{th}} \) denote the initial change point, and \( P^{(0)}_1 = H_K, P^{(0)}_0 = [n] \setminus H_K \) denote the initial sets of indices; both are defined in Step 3a.

Now, suppose that \( W^{(\ell)}_{\text{next}} \) is the last change point computed in the while loop, before the stopping rule of Step 3b is invoked. That is, one of the following cases takes place:

(i) There does not exist \( i \in P^{(\ell)}_1, j \in P^{(\ell)}_0 \) such that \( w_i < w_j \). In this case, we exit the while loop before the start of the \((\ell + 1)\)th iteration.

(ii) Within the \( \ell \)th iteration, Step 3(b)i of Algorithm 1, after making the following update:

\[
(i^*, j^*) \in \arg \min_{i \in P^{(\ell-1)}_1, j \in P^{(\ell-1)}_0 \atop w_i < w_j} \frac{c_j - c_i}{w_j - w_i},
\]

we have \( \frac{c_{j^*} - c_{i^*}}{w_{j^*} - w_{i^*}} > \frac{1}{1 + W^{(\ell)}_{\text{next}}} \). That is, we hit the stopping rule in Step 3(b)iii.

In the following, we will show that by the time we reach our last change point \( W^{(\ell)}_{\text{next}} \), we have already reached \( W_{\text{max}} \). That is, \( W^{(\ell)}_{\text{next}} \geq W_{\text{max}} \).

For Case (i), contrary to our claim, assume that \( W^{(\ell)}_{\text{next}} < W_{\text{max}} \). Then, as we will show in Part 5 of the proof, at capacity level \( W^{(\ell)}_{\text{next}} \), we have that \( \mathcal{P}(W^{(\ell)}_{\text{next}}) = \{P^{(\ell)}_1, (0, 0), P^{(\ell)}_0\} \). We also have \( w(P^{(\ell)}_1) = W^{(\ell)}_{\text{next}} \).

Observe that in Step 3 of Algorithm 1, in every iteration of the while loop, the set \( P_1 \) always contain \( K \) items. Hence, if there does not exist \( i \in P^{(\ell)}_1, j \in P^{(\ell)}_0 \) such that \( w_i < w_j \), this means that \( P^{(\ell)}_1 \) already contains the \( K \) items with the highest weights. Since we cannot find a set of \( K \) items with total weight higher than \( w(P^{(\ell)}_1) = W^{(\ell)}_{\text{next}} \), for any \( W' > W^{(\ell)}_{\text{next}} \), the capacity constraint must be nonbinding for Problem KP-RELAX\((W', K)\), which contradicts \( W_{\text{max}} > W^{(\ell)}_{\text{next}} \). By proof of contradiction, we thus have \( W^{(\ell)}_{\text{next}} \geq W_{\text{max}} \).

For Case (ii), we first remark that if the previous change point \( W^{(\ell-1)}_{\text{next}} > W_{\text{max}} \), we would trivially have \( W^{(\ell)}_{\text{next}} > W_{\text{max}} \). Hence, in the following lemma, we assume that \( W^{(\ell-1)}_{\text{next}} \leq W_{\text{max}} \). Again by Part 5 of the proof, we would have \( \mathcal{P}(W^{(\ell)}_{\text{next}}) = \{P^{(\ell)}_1, (0, 0), P^{(\ell)}_0\} \). This allows us to invoke Lemma 4 stated below, which suggests that if we hit the stopping rule in Step 3(b)iii, we have \( W^{(\ell)}_{\text{next}} > W_{\text{max}} \).

**Lemma 4.** Consider the change point \( W^{(\ell-1)}_{\text{next}} \) and the two sets of indices \( P^{(\ell-1)}_1, P^{(\ell-1)}_0 \) computed in the \((\ell - 1)\)th iteration in Step 3b of Algorithm 1. Let \( W^{(\ell)}_{\text{next}} \) be the change point computed in the \( \ell \)th iteration, and let \( i^*, j^* \) be the two indices defined in Equation (20). Assume that \( W^{(\ell)}_{\text{next}} \leq W_{\text{max}} \) and \( \mathcal{P}(W^{(\ell)}_{\text{next}}) = \{P^{(\ell-1)}_1, (0, 0), P^{(\ell-1)}_0\} \). Then, if \( \frac{c_{j^*} - c_{i^*}}{w_{j^*} - w_{i^*}} > \frac{1}{1 + W^{(\ell)}_{\text{next}}} \), we have \( W^{(\ell)}_{\text{next}} > W_{\text{max}} \).
Part 4: if $W^* \in I_{\text{low}}$, a 1/2-approx. solution is found in Step 2 of Algorithm 1. This part uses the following lemma:

**Lemma 5.** Consider the relaxed knapsack problem (kp-relax($W, K$)). If the capacity level satisfies $W \in I_{\text{low}} = [0, w(H_K)]$, where $H_K \subset [n]$ is the set of $K$ items in $[n]$ with the highest utility-to-weight ratios, the optimal solution to (kp-relax($W, K$)) is given by filling up the knapsack with items with positive utilities in the descending order of the utility-to-weight ratio until the capacity is reached.

Suppose that $W^* \in I_{\text{low}}$. Lemma 5 shows that an optimal basic feasible solution $x^*(W^*)$ to the relaxed knapsack problem (kp-relax($W^*, K$)) is simply filling up the knapsack with the items with the highest utility-to-weight ratios until we reach capacity $W^*$. Recall that the utility-to-weight ratio of item $i \in [n]$ in Problem (kp-relax($W, K$)) is

$$\frac{u_i(W)}{w_i} = \frac{1}{1 + W} - \frac{c_i}{w_i},$$

and hence, the order of utility-to-weight ratio of items does not depend on the capacity level $W$. This means that the profile $P(W^*)$ of the optimal solution must take one of the following forms:

(i) $P_1(W^*) = H_j$ for some $j < K$, and $P_j(W^*) = (0, 0)$. This means that the knapsack is precisely filled by the $j$ items in $[n]$ with the highest utility-to-space ratios.

(ii) $P_j(W^*) = H_{j-1}$ for some $j \in [K]$, and $P_j(W^*) = (h_j, 0)$. This means that the knapsack is filled by the $(j-1)$ items in $[n]$ with the highest utility-to-space ratios, and partially filled by the item $h_j$, i.e., the item with the $j$-th highest utility-to-space ratio.

Given the form of the profile $P(W^*)$, we know from Lemma 1 that there must exist a 1/2 approximate feasible assortment $S_{W^*}$ to Problem (kp-relax($W^*, K$)) in the following collection: $\{\{j\} : j \in [n]\} \cup \{H_j : j \in [K-1]\}$. Observe that in Step 1 of Algorithm 1, we add $\{j\}$ to $C$ for all $j \in [n]$. In Step 2 of Algorithm 1, we add $\{H_j : j \in [K-1]\}$ to $C$. Therefore,

$$S_{W^*} \subseteq \{\{j\} : j \in [n]\} \cup \{H_j : j \in [K-1]\} \subseteq C,$$

where $S_{W^*} \subseteq C$ is a 1/2-approx. assortment for Problem kp-relax($W^*, K$).

**Part 5:** if $W^* \in I_{\text{high}}$, a 1/2-approx. solution is found in Step 3 of Algorithm 1. Recall from Part 1 of the proof that $W^* \leq W_{\text{max}}$. In the following, suppose $W^* \in I_{\text{high}} \cap [0, W_{\text{max}}] = [W_{\text{th}}, W_{\text{max}}]$. We will show that an 1/2 approximate solution is included in $C$.

Recall from Part 3 of the proof, we make the following definitions: let $W_{\text{next}}^{(k)}$ be the change point updated in the $k$th iteration of the while loop, and $P_1^{(k)}, P_0^{(k)}$ be the sets of indices updated in the $k$th iteration. Additionally, $W_{\text{next}}^{(0)} = W_{\text{th}}$ denotes the initial change point, and $P_1^{(0)} = H_K, P_0^{(0)} = [n] \setminus H_K$ denote the initial sets of indices.

Let us start by showing the following statement: suppose that at the $k$th iteration, for some $k \geq 0$, Algorithm 1 has updated the change point to be $W_{\text{next}}^{(k)}$ and the two sets to be $P_1^{(k)}, P_0^{(k)}$, then, as long as $W_{\text{next}}^{(k)} \leq W_{\text{max}}$, the optimal basic solution at $W_{\text{next}}^{(k)}$ has profile $P(W_{\text{next}}^{(k)}) = \{P_1^{(k)}, (0, 0), P_0^{(k)}\}$. We will prove this statement via induction:
• **Base step** \((k = 0)\): Given that \(u_j(W_{th}) > 0\) for all \(j \in H_K\), we know from Lemma 5 that the optimal basic solution \(x^*(W_{th})\) for Problem \(KPRELAX(W_{th}, K)\) has the profile \(P_1(W_{th}) = H_K\), \(P_1(W_{th}) = (0, 0)\), and \(P_0(W_{th}) = [n] \setminus H_K\). In Step 3a of Algorithm 1, we let \(W_{next}^{(0)} = W_{th}\) be our initial change point, and initialize the two sets to be \(P_1^{(0)} = H_K\) and \(P_0^{(0)} = [n] \setminus H_K\). Hence the base case holds.

• **Inductive step**: Suppose that \(W_{next}^{(k)}, P_1^{(k)}, P_0^{(k)}\) and \(W_{next}^{(k+1)}, P_1^{(k+1)}, P_0^{(k+1)}\) are the quantities at the \(k\)th and \((k+1)\)th iteration respectively. Let us assume that \(W_{next}^{(k)} < W_{max}^{(k+1)}\). In addition, assume that the aforementioned statement holds at \(W_{next}^{(k)}\), i.e., the optimal solution at \(W_{next}^{(k)}\) has profile \(P(W_{next}^{(k)}) = \{P_1^{(k)}, (0, 0), P_0^{(k)}\}\). Within the \((k+1)\)th iteration, to update the quantities, in Step 3(b)i and 3(b)ii of Algorithm 1, we first solve the following optimization problem \(i^{*}, j^{*} \in \arg \min_{i \in P_1^{(k)}, j \in P_0^{(k)}} \frac{c_j - c_i}{w_j - w_i}\), and then make the following updates: \(W_{next}^{(k+1)} = W_{next}^{(k)} + w_{i^*} + w_{j^*}, P_1^{(k+1)} = P_1^{(k)} \cup \{j^*\}\) and \(P_0^{(k+1)} = P_0^{(k)} \cup \{i^*\}\). To show the result, we make use of the following lemma:

**Lemma 6.** Suppose that at capacity level \(W\), the capacity constraint is binding and the relaxed knapsack problem \((KPRELAX(W, K))\) has a degenerate, integer optimal basic solution \(x^*\), with profile \(P(W) = \{P_1^{(k)}, (0, 0), P_0^{(k)}\}\). Let \((i^*, j^*) \in \arg \min_{i \in P_1^{(k)}, j \in P_0^{(k)}} \frac{c_j - c_i}{w_j - w_i}\). For any \(\eta \in [0, w_{i^*} - w_{j^*}]\) such that the capacity constraint is binding, the problem \((KPRELAX(W + \eta, K))\) has an optimal basic solution \(x\) such that \(x_k = 1\) for any \(k \in P_1 \setminus \{i^*\}\), \(x_k = 0\) for any \(k \in P_0 \setminus \{j^*\}\), and

\[
x_k = \begin{cases} 1 - \frac{\eta}{w_{j^*} - w_{i^*}} & k = i^* \\ \frac{\eta}{w_{j^*} - w_{i^*}} & k = j^* \end{cases}
\]

Note that the proof of the inductive step is completed by invoking Lemma 6 with \(W = W_{next}^{(k)}\) and \(\eta = w_{j^*} - w_{i^*}\). Recall that \(W_{next}^{(k+1)} = W_{next}^{(k)} + w_{j^*} - w_{i^*}\). Further note that at both \(W_{next}^{(k)}\) and \(W_{next}^{(k+1)}\), the capacity constraint is binding as we are showing the induction step for any change points less than \(W_{max}\).

The base step and the inductive step together prove the statement that at any given iteration of Step 3b, as long as the change point \(W_{next} \leq W_{max}\), the optimal basic solution at \(W_{next}\) has profile \(P(W_{next}) = \{P_1^{(k)}, (0, 0), P_0^{(k)}\}\).

Now, suppose that throughout Algorithm 1, we have the following change points: \(W_{th} = W_{next}^{(0)} < W_{next}^{(1)} < \cdots < W_{next}^{(l)}\), where \(W_{next}^{(l)}\) is the last change point computed by Algorithm 1 before the stopping rule in Step 3b is invoked. We have shown, in part 3 of the proof, that \(W^* \leq W_{next}^{(l)}\).

Therefore, we must have \(W^*\) lie in between two consecutive change points, i.e. \(W^* \in [W_{next}^{(k)}, W_{next}^{(k+1)}]\) for some \(0 \leq k < l\). Since \(W^* \leq W_{max}\), as shown in Step 1 of the proof, we must also have \(W_{next}^{(k)} \leq W_{max}\); hence, our inductive statement holds for the \(k\)th iteration. At the \(k\)th iteration of Step 3b, we have updated the change point to be \(W_{next}^{(k)}\), and the two sets to be \(P_1^{(k)}, P_0^{(k)}\). By the inductive statement, we have that \(P(W_{next}^{(k)}) = \{P_1^{(k)}, (0, 0), P_0^{(k)}\}\). Since \(W^* \in [W_{next}^{(k)}, W_{next}^{(k+1)}]\) and that the capacity constraint is binding at \(W^*\), by Lemma 6, we have:

(i) if \(W^* = W_{next}^{(k)}\) for some \(k \in \{0, \ldots, l\}\), the profile of \(x^*(W^*)\) is \(P(W^*) = \{P_1^{(k)}, (0, 0), P_0^{(k)}\}\).

(ii) if \(W^* \in (W_{next}^{(k)}, W_{next}^{(k+1)})\) for some \(k \in \{0, \ldots, l - 1\}\), the profile of \(x^*(W^*)\) is \(P(W^*) = \{P_1^{(k)} \setminus \{i^*\}, (i^*, j^*), P_0^{(k)} \setminus \{j^*\}\}\).
Given the form of the profile \( P(W^*) \), we know from Lemma 1 that either \( P_1^{(k)} \) or \( \{j^*\} \) is an 1/2-approx. feasible solution \( S_{W^*} \) to Problem kp-relax\((W^*, K)\). Observe that in Step 3(b)iii of the \( k \)th iteration, we have not hit the stopping rule and added \( P_1^{(k)} \) to the collection \( C \). In Step 1, we have added \( \{j\} \) to \( C \) for all \( j \in [n] \). Therefore, if \( W^* \in I_{\text{high}} \), we again have \( S_{W^*} \in C \).

Overall, parts 1–5 together imply that whether \( W^* \in I_{\text{low}} \) or \( W^* \in I_{\text{high}} \), Algorithm 1 would add a 1/2 approximate feasible assortment \( S_{W^*} \) to the collection \( C \). Hence, Claim 1 holds.

Finally, we remark that the proof for Claim 1 above in fact applies to all capacity levels \( W \leq W_{\text{max}} \), and can yield a more general claim as follows:

**Claim 2.** For any \( W \leq W_{\text{max}} \), where \( W_{\text{max}} \) is the maximum capacity level under which the capacity constraint is binding, there exists an assortment \( S_W \) in the collection of assortments \( C \) maintained in Algorithm 1 such that \( S_W \) is an integer 1/2 approximate feasible solution for Problem kp-relax\((W, K)\).

**Segment 2.** We now comment on the running time of Algorithm 1. In Step 1, adding \( n \) singletons to \( C \) takes \( O(n) \), and ranking the items by their utility-to-weight ratios takes \( O(n \log n) \). In Step 2, adding \( n \) assortments to \( C \) and checking the signs of the utilities takes \( O(n) \). For Step 3, we can pre-compute and sort \( \frac{w_i - c_i}{w_j - w_i} \) for all \( i, j \in [n] \) such that \( w_i < w_j \). The pre-computing step takes \( O(n^2) \), and sorting takes \( O(n^2 \log n) \). Since the values \( \{\frac{w_i - c_i}{w_j - w_i} : i, j \in [n], w_i < w_j\} \) have been pre-computed and sorted, choosing \((i^*, j^*)\) that minimizes \( \frac{w_i - c_i}{w_j - w_i} \) and updating \( P_1, P_0, W_{\text{next}} \) both take \( O(1) \). For any sub-interval that follows, we make \( O(n) \) changes to the pairs of \((i, j)\) that we consider, since we simply exchange one element in \( P_1 \) with another element in \( P_0 \). After we determine the subset \( \{(i, j) : i \in P_1, i \in P_0, w_i < w_j\} \) for the current sub-interval, choosing \((i^*, j^*)\) and updating \( P_1, P_0, W_{\text{next}} \) again takes \( O(1) \).

We further remark that the adaptive partition of \( I_{\text{high}} \) would divide \( I_{\text{high}} \) into at most \( O(nK) \) sub-intervals. This is because at each iteration of Step 3b, we need to swap item \( i \in P_1 \) and \( j \in P_0 \) such that \( w_i < w_j \); that is, at each iteration, we need to replace one item in \( P_1 \) with another item of higher weight. Since there are \( K \) items in \( P_1 \), we can perform at most \((n - 1) + (n - 2) + \cdots + (n - K) = O(nK) \) such replacements. In the first sub-interval that starts at \( W_{\text{hi}} \), we consider \( K(n - K) \) pairs of \((i, j)\), where \( i \in P_1(W_{\text{hi}}) \) and \( j \in P_0(W_{\text{hi}}) \), which takes \( O(nK) \). Overall, the total time taken in the adaptive partitioning would be \( O(K(n - K) + nK \cdot n) = O(n^2K) \). Finally, since \( O(n) \) assortments are added in Step 1, \( O(n) \) assortments are added in Step 2, and \( O(nK) \) assortments are added in Step 3, there are at most \( O(nK) \) assortments in the collection \( C \). The time taken to find the assortment that maximizes ms-cost in Step 4 is thus \( O(nK) \). The overall complexity of Algorithm 1 is \( O(n \log n + n + n^2 \log n + n^2K + nK) = O(n^2(\log n + K)) \).

**F.1. Proof of Lemma 2**

Consider a capacity level \( \tilde{W} > 0 \), and suppose that the capacity constraint is non-binding at the optimal basic solution \( x^*(\tilde{W}) \) to kp-relax\((\tilde{W}, K)\). That is, \( \sum_{i \in [n]} w_i x_i^*(\tilde{W}) < \tilde{W} \). We would like to show that for any \( W > \tilde{W}, \) we have \( \sum_{i \in [n]} w_i x_i^*(W) < W \). Contrary to our claim, suppose that at capacity level \( \tilde{W} + \eta \) for some \( \eta > 0 \), the capacity constraint is binding for the optimal solution \( x^*(\tilde{W} + \eta) \). That is, \( \sum_{i \in [n]} w_i x_i^*(\tilde{W} + \eta) = \tilde{W} + \eta \). We would show the contradiction by showing that \( x^*(\tilde{W} + \eta) \) is not optimal and finding another solution that yields a higher objective.
To do that, we first consider the following auxiliary relaxed problem:

\[
\max_{x \in [0, 1]^n} \sum_{i \in [n]} \left( \frac{w_i}{1+W} - c_i \right) x_i
\]

subject to

\[
\sum_{i \in [n]} w_i x_i \leq \tilde{W} + \eta
\]

\[
\sum_{i \in [n]} x_i \leq K
\]

(22)

Note that the above relaxed problem only differs from \text{KP-relax}(\tilde{W}, K) in the capacity constraint. Since the capacity constraint is non-binding for Problem \text{KP-relax}(\tilde{W}, K), we have that \(x^*(\tilde{W})\) is also the optimal solution to Problem (22). This is because both feasibility and optimality conditions remain unchanged. We thus have

\[
\sum_{i \in [n]} \left( \frac{w_i}{1+W} - c_i \right) x_i^*(\tilde{W}) \geq \sum_{i \in [n]} \left( \frac{w_i}{1+W} - c_i \right) x_i^*(\tilde{W} + \eta).
\]

(23)

This then gives

\[
\sum_{i \in [n]} \left( \frac{w_i}{1+W + \eta} - c_i \right) x_i^*(\tilde{W}) = \sum_{i \in [n]} \left( \frac{w_i}{1+W} - c_i \right) x_i^*(\tilde{W}) - \left( \frac{1}{1+W} - \frac{1}{1+W + \eta} \right) \sum_{i \in [n]} w_i x_i^*(\tilde{W})
\]

\[
\geq \sum_{i \in [n]} \left( \frac{w_i}{1+W} - c_i \right) x_i^*(\tilde{W} + \eta) - \left( \frac{1}{1+W} - \frac{1}{1+W + \eta} \right) \sum_{i \in [n]} w_i x_i^*(\tilde{W})
\]

\[
> \sum_{i \in [n]} \left( \frac{w_i}{1+W} - c_i \right) x_i^*(\tilde{W} + \eta) - \left( \frac{1}{1+W} - \frac{1}{1+W + \eta} \right) \sum_{i \in [n]} w_i x_i^*(\tilde{W} + \eta)
\]

\[
= \sum_{i \in [n]} \left( \frac{w_i}{1+W + \eta} - c_i \right) x_i^*(\tilde{W} + \eta),
\]

where the first inequality follows from Equation (23), and the second inequality follows from the fact that

\[
\sum_{i \in [n]} w_i x_i^*(\tilde{W}) < \tilde{W} < \tilde{W} + \eta = \sum_{i \in [n]} w_i x_i^*(\tilde{W} + \eta).
\]

However, this contradicts the fact that \(x^*(\tilde{W} + \eta)\) is an optimal solution for the problem \text{(KP-relax}(W^* + \eta, K)). We thus show that the capacity constraint must be non-binding for all \(W \geq \tilde{W}\).

F.2. Proof of Lemma 3

Let \(W^* \in \arg \max_{W \in [0, \infty)} \text{KP-relax}(W, K)\) and define \(x^* = x^*(W^*)\) as the optimal basic solution to Problem \text{(KP-relax}(W^*, K)). We would like to show that the capacity constraint in Problem \text{(KP-relax}(W^*, K)) is binding. That is, \(\sum_{i \in [n]} w_i x_i^* = W^*\). Contrary to our claim, suppose that \(\sum_{i \in [n]} w_i x_i^* = W' < W^*\). However, this then implies

\[
\text{KP-relax}(W', K) \geq \sum_{i \in [n]} \left( \frac{w_i}{1+W'} - c_i \right) x_i^* > \sum_{i \in [n]} \left( \frac{w_i}{1+W^*} - c_i \right) x_i^* = \text{KP-relax}(W^*, K),
\]

(24)

where the first inequality holds because \(\sum_{i \in [n]} w_i x_i^* = W' < W^*\) and hence \(x^*\) is a feasible solution to the Problem \text{(KP-relax}(W', K)). However, the inequality in (24) contradicts that fact that \(\max_{W \in [0, \infty)} \text{KP-relax}(W, K) = \text{KP-relax}(W^*, K)\). The statement of this Lemma thus holds.
F.3. Proof of Lemma 4

We first note that for any $i, j \in [n]$, we have the following equivalence:

$$\frac{c_j - c_i}{w_j - w_i} > \frac{1}{1 + W_{next}^{(\ell)}} \iff \frac{w_i}{1 + W_{next}^{(\ell)}} - c_i > \frac{w_j}{1 + W_{next}^{(\ell)}} - c_j \iff u_i(W_{next}^{(\ell)}) > u_j(W_{next}^{(\ell)}).$$

Now, if $\frac{c_j - c_i}{w_j - w_i} > \frac{1}{1 + W_{next}^{(\ell)}}$, where $(i^*, j^*) \in \arg\min_{i,j \in \{1,2,\ldots\}, j \in P_0^{(\ell-1)} - w_i < w_j}$ with $w_i < w_j$. That is, for Problem (kp-relax($W_{next}^{(\ell)}, K)$), the $K$ items with the highest utilities $u_i(W_{next}^{(\ell)})$ are all included in $P_1^{(\ell-1)}$. Hence, the optimal solution to Problem (kp-relax($W_{next}^{(\ell)}, K$)) is $S^* = \{i \in P_1^{(\ell-1)} : u_i(W_{next}^{(\ell)}) \geq 0\}$. The set $S^*$ is also a feasible solution to Problem (kp-relax($W_{next}^{(\ell)}, K$)). This is because (i) by assumption $W_{next}^{(\ell-1)} \leq W_{max}$, and (ii) $w(S^*) \leq w(P_1^{(\ell-1)}) = W_{next}^{(\ell-1)} < W_{next}^{(\ell)}$. The last chain of inequalities also shows that the capacity constraint is non-binding for Problem (kp-relax($W_{next}^{(\ell)}, K$)), and thus, by Lemma 2, we have $W_{next}^{(\ell)} > W_{max}$. ■

F.4. Proof of Lemma 5

Consider the relaxed knapsack problem without the cardinality constraint:

$$\max_{x \in [0,1]^n} \sum_{i=1}^n \left( \frac{w_i}{1+W} - c_i \right)x_i$$

s.t. $\sum_{i=1}^n w_ix_i \leq W$  \hspace{1cm} (25)

It is known that the optimal solution to Problem (25) is given by filling up the knapsack in the descending order of the utility-to-weight ratio until the capacity is reached. Note that since $W \leq w(H_K)$, this optimal solution also satisfies $|\sum_{i \in [n]} x_i| \leq K$ and is thus feasible for Problem (kp-relax($W, K$)) as well. Since kp-relax($W, K$) is always upper bounded by the optimal objective of (25), we have the optimal solution to Problem (25) is also optimal for Problem (kp-relax($W, K$)). ■

F.5. Proof of Lemma 6

The result trivially holds when $\eta = 0$. For any fixed $\eta \in (0, w_j - w_i]$, we can choose $\eta' \in (0, \min\{\eta, \min_{i \neq j} |w_i - w_j|\})$ such that $W + \eta' \neq \sum_{i \in S} w_i$ for any $S \subset [n], |S| = K$. Note that such $\eta'$ always exists. (Recall that we assume $w_i \neq w_j$ for all $i \neq j$. Here, the set of values that $\eta'$ cannot take is discrete, so there always exists $\eta'$ in the nonempty interval $(0, \min\{\eta, \min_{i \neq j} |w_i - w_j|\})$ that satisfies this condition.) Since the capacity constraint is binding for Problem (kp-relax($W + \eta, K$)), by Lemma 2, it is also binding for Problem (kp-relax($W + \eta', K$)). Our proof consists of two parts: (i) we first solve for the optimal solution at $W + \eta'$, (ii) we next show that the optimal solutions at $W + \eta'$ and $W + \eta$ share the same profile, which allows us to derive $\mathbf{x}^*(W + \eta)$.

First, let us solve for the optimal solution at $W + \eta'$ using the following lemma:

**Lemma 7.** Suppose that at capacity level $W \geq w(H_K)$, the capacity constraint is binding and the relaxed knapsack problem (kp-relax($W, K$)) has a degenerate, integer optimal basic solution $\mathbf{x}^*$, with profile $\mathcal{P}(W) = \{P_1, (0,0), P_0\}$. For any $0 < \eta' \leq \min_{i \neq j} |w_i - w_j|$ such that $W + \eta' \neq \sum_{i \in S} w_i$ for any $S \subset [n], |S| = K$, if the
capacity constraint is binding at capacity level $W + \eta'$, the problem (KP-RELAX$(W + \eta', K)$) has an optimal basic solution, where $x_k = 1$ for any $k \in P_1 \setminus \{i^*\}$, $x_k = 0$ for any $k \in P_0 \setminus \{j^*\}$, and

$$x_k = \begin{cases} 1 - \frac{c_{j^*} - c_{i^*}}{w_{j^*} - w_{i^*}} & k = i^* \\ \frac{w_k - w_{j^*}}{w_{j^*} - w_{i^*}} & k = j^* \end{cases}. \quad (26)$$

Here $(i^*, j^*) \in \arg \min_{i \in P_1, j \in P_0} \left\{ \frac{c_j - c_i}{w_j - w_i} \right\}$.

Using Lemma 7, we can solve for the optimal basic solution $x^*(W + \eta')$, which takes the form in Equation (26). The solution has the profile $P(W + \eta') = \{P_1 \setminus \{i^*\}, (i^*, j^*), P_0 \setminus \{j^*\}\}$.

Now, consider $\eta \geq \eta'$, which falls into the interval $(0, w_j - w_i)$. We show that the optimal solutions at $W + \eta'$ and $W + \eta$ share the same profile, and as soon as we know the profile $P(W + \eta)$, we can solve for the optimal solution $x^*(W + \eta)$ by solving a simple linear system defined by the capacity and cardinality constraints. This is done in the proof of the following lemma:

**Lemma 8.** Suppose that at capacity level $\tilde{W} > w(H_K)$, an optimal basic solution $x^*(\tilde{W})$ to Problem (KP-RELAX$(\tilde{W}, K)$) has profile $P(\tilde{W}) = \{P_1 \setminus \{i\}, (i, j), P_0 \setminus \{j\}\}$. Then, for any $\tilde{W} \in [w(P_1), w(P_1) - w_i + w_j]$ such that $\frac{c_{j^*} - c_i}{w_{j^*} - w_{i^*}} \leq \frac{1}{1 + W}$, we have an optimal basic solution $x$ to Problem (KP-RELAX$(\tilde{W}, K)$) such that $x_k = 1$ for any $k \in P_1 \setminus \{i\}$, $x_k = 0$ for any $k \in P_0 \setminus \{j\}$, and

$$x_k = \begin{cases} \frac{W - w(P_1) + w_i - w_j}{w_i - w_j} & k = i \\ \frac{W - w(P_1)}{w_j - w_i} & k = j \end{cases}. \quad (27)$$

We now invoke Lemma 8 by taking $\tilde{W} = W + \eta'$ and $\tilde{W} = W + \eta$. Recall that we have assumed that the capacity constraint is binding at $W + \eta$, and by Lemma 11, we have $\frac{c_{j^*} - c_i}{w_{j^*} - w_{i^*}} \leq \frac{1}{1 + W + \eta}$. Then, given the profile $P(W + \eta') = \{P_1 \setminus \{i^*\}, (i^*, j^*), P_0 \setminus \{j^*\}\}$, we can apply Lemma 8 to get the optimal solution $x$ to Problem (KP-RELAX$(W + \eta, K)$) such that $x_k = 1$ for any $k \in P_1 \setminus \{i^*\}$, $x_k = 0$ for any $k \in P_0 \setminus \{j^*\}$,

$$x_{i^*} = \frac{W + \eta - W + w_j - w_{j^*}}{w_{j^*} - w_{i^*}} = 1 - \frac{\eta}{w_{j^*} - w_{i^*}},$$

and

$$x_{j^*} = \frac{W + \eta - W}{w_{j^*} - w_{i^*}} = \frac{\eta}{w_{j^*} - w_{i^*}},$$

which is the desired form. ■

**F.6. Proof of Lemma 7.**

Suppose that at capacity level $W$, the capacity constraint is binding and the relaxed knapsack problem (KP-RELAX$(W, K)$) has a degenerate, integer optimal basic solution $x^*$, with profile $P(W) = \{P_1, (0, 0), P_0\}$. Fix $0 < \eta' \leq \min_{i \neq j} |w_i - w_j|$ such that $W + \eta' \neq \sum_{i \in S} w_i$ for any $S \subset [n], |S| = K$, and suppose that the capacity constraint is binding for the problem KP-RELAX$(W + \eta', K)$. Let us first consider the following auxiliary relaxed problem, in which we only increase the capacity level from $W$ to $W + \eta'$, without changing the utility of each item:

$$\max_{x \in [0, 1]^n, x_1 + x_2 \geq 0} \sum_{i \in [n]} \left( \frac{w_i}{1 + W} - c_i \right)x_i$$

s.t. $\sum_{i \in [n]} w_ix_i + s_1 = W + \eta'$

$$\sum_{i \in [n]} x_i + s_2 = K \quad (28)$$
Note that in the above optimization problem, we introduce \( s_1, s_2 \) to turn inequality constraints into equality constraints. Our proof consists of two parts. In Part 1, we find an optimal basic solution to the auxiliary problem in (28). To do that, we apply ideas from the dual simplex method, and show that (i) the optimal basis \( \{i^*, j^*\} \) has the following properties: (i) \( i^* \in P_1, j^* \in P_0 \) and \( w_{i^*} < w_{j^*} \), and (ii) \( (i^*, j^*) \in \arg \min_{i,j \in P_0} \{ \frac{c_j - c_i}{w_j - w_i} \} \). In Part 2, we show that the optimal solution to the auxiliary problem in (28) is also an optimal basic solution to Problem (KP-RELAX(\( W + \eta', K \))). Our proof below relies on notions such as optimality and feasibility conditions, reduced costs, the simplex and dual simplex methods, which are defined and described in more details in Appendix J.

**Part 1: Find an optimal basic solution \( \hat{x} \) to the auxiliary problem.** Observe that when we increase the capacity level from \( W \) to \( W + \eta' \), the optimality condition still holds, since the reduced cost associated with each variable is unaffected. However, the feasibility condition does not necessarily hold for the original basic variables. To look for a new optimal basis for the auxiliary problem, we can apply the dual simplex method. By design of the dual simplex method, throughout its execution, the optimality condition always hold (i.e., the reduced costs for all non-basic variables at their lower bounds are non-positive; the reduced costs for all non-basic variables at their upper bounds are non-negative; the reduced costs for all basic variables are zero). The dual simplex method terminates when the feasibility condition is also satisfied; see Appendix J and Bertsimas and Tsitsiklis (1997) for details.

Before applying the dual simplex method to find a new optimal basis, we first make a few remarks about the optimal solution to Problem (28), which we denote as \( \hat{x} \). By Lemma 9 stated below, we have that \( \hat{x} \) must have either two fractional components or have all integer variables. Via a similar argument as in the proof of Lemma 2, we can show that since the capacity constraint is binding for Problem KP-RELAX(\( W + \eta', K \)), the capacity constraint must also be binding for Problem (28); that is, \( \sum_{i \in [n]} w_i \hat{x}_i = W + \eta' \). Then, by Lemma 10 stated below, we have that the cardinality constraint is also binding for Problem (28); that is, \( \sum_{i \in [n]} \hat{x}_i = K \).

Since \( W + \eta' \neq \sum_{i \in S} w_i \) for any \( S \subset [n], |S| = K \), there does not exist an integer solution \( \hat{x} \) that can satisfy the above conditions. Hence, the optimal basic solution \( \hat{x} \) must have two fractional components, \( \hat{x}_{i^*} \) and \( \hat{x}_{j^*} \), which also serve as the basic variables. In the following, we look for this optimal basis \( \{i^*, j^*\} \) with the help of the dual simplex method.

**Lemma 9.** Consider the knapsack problem stated in (28). If \( W \geq w(H_K) \), an optimal basic solution \( x^* \) to Problem (28) either has exactly two fractional components, or all variables take integer values.

**Lemma 10.** Consider the knapsack problem stated in (28). If \( W \geq w(H_K) \) and the capacity constraint is binding, then the cardinality is also binding.

Suppose that we apply the dual simplex method to solving Problem (28). The dual simplex method works in the following way. At each iteration, it updates the basis \( \{i,j\} \) by removing one index out of the basis and adding a different index into the basis. It then computes the values of the basic variables \( x_i, x_j \) to ensure that the capacity and cardinality constraints are tight. The non-basic variables \( \{x_k : k \neq i,j\} \) would remain at their respective lower/upper bounds. The dual simplex method then checks whether the feasibility condition is satisfied, i.e., \( x_i, x_j \in [0, 1] \). If it is satisfied, the dual simplex terminates; otherwise, it starts a new
From the case discussion above, we have seen that the dual simplex method must terminate at an optimal basis \( \{i^*, j^*\} \), we must have \( i^* \in P_1, j^* \in P_0 \) and \( w_{i^*} < w_{j^*} \). Recall that at capacity level \( W \), we assumed that (KP-RELAX\((W, K)\)) has a degenerate, integer optimal basic solution \( x^* \), with profile \( P(W) = \{P_1, (0, 0), P_0\} \). Further recall that the dual simplex method terminates when the feasibility conditions hold for our chosen basis. We can check the feasibility conditions for the following three cases:

1. Case 1: Both \( i^* \) and \( j^* \) are from \( P_0 \). If we solve for \( \hat{x}_{i^*}, \hat{x}_{j^*} \), using the binding capacity and cardinality constraints, i.e.

\[
 w_{i^*} \hat{x}_{i^*} + w_{j^*} \hat{x}_{j^*} = \eta' \quad \text{and} \quad \hat{x}_{i^*} + \hat{x}_{j^*} = 0,
\]

we have

\[
 \hat{x}_{i^*} = -\frac{\eta'}{w_{j^*} - w_{i^*}} \quad \text{and} \quad \hat{x}_{j^*} = \frac{\eta'}{w_{j^*} - w_{i^*}}.
\]

Clearly, here the feasibility condition is again not satisfied because one of the variables would exceed one. Hence, \( \hat{x} \) can not be the optimal solution.

2. Case 2: Both \( i^* \) and \( j^* \) are from \( P_1 \). If we solve for \( \hat{x}_{i^*}, \hat{x}_{j^*} \), using the binding capacity and cardinality constraints, i.e.

\[
 w_{i^*} \hat{x}_{i^*} + w_{j^*} \hat{x}_{j^*} = w_{i^*} + w_{j^*} + \eta' \quad \text{and} \quad \hat{x}_{i^*} + \hat{x}_{j^*} = 2,
\]

we have

\[
 \hat{x}_{i^*} = 1 - \frac{\eta'}{w_{j^*} - w_{i^*}} \quad \text{and} \quad \hat{x}_{j^*} = 1 + \frac{\eta'}{w_{j^*} - w_{i^*}}.
\]

Clearly, here the feasibility condition is again not satisfied because one of the variables would exceed one. Hence, \( \hat{x} \) cannot be the optimal solution.

3. Case 3: One variable is from \( P_1 \) and the other is from \( P_0 \). (WLOG, let \( i^* \in P_1 \) and \( j^* \in P_0 \).) If we solve for \( \hat{x}_{i^*}, \hat{x}_{j^*} \), using the binding capacity and cardinality constraints, i.e.

\[
 w_{i^*} \hat{x}_{i^*} + w_{j^*} \hat{x}_{j^*} = w_{i^*} + \eta' \quad \text{and} \quad \hat{x}_{i^*} + \hat{x}_{j^*} = 1,
\]

we get

\[
 \hat{x}_{i^*} = 1 - \frac{\eta'}{w_{j^*} - w_{i^*}} \quad \text{and} \quad \hat{x}_{j^*} = \frac{\eta'}{w_{j^*} - w_{i^*}}.
\]

Here, the solution \( \hat{x} \) would be feasible if and only if \( w_{j^*} - w_{i^*} > 0 \).

From the case discussion above, we have seen that the dual simplex method must terminate at an optimal basis \( \{i^*, j^*\} \) that satisfies \( i^* \in P_1, j^* \in P_0 \) and \( w_{i^*} < w_{j^*} \). In particular, in the optimal basic solution \( \hat{x} \) returned by the dual simplex method, the two basic variables would take the value \( \hat{x}_{i^*} = 1 - \frac{\eta'}{w_{j^*} - w_{i^*}} \) and \( \hat{x}_{j^*} = \frac{\eta'}{w_{j^*} - w_{i^*}} \), while the other non-basic variables would remain unchanged form the optimal solution to problem (KP-RELAX\((W, K)\)), i.e., \( \hat{x}_k = x^*(k) \).

To determine \( (i^*, j^*) \), we further note that if we choose \( i \in P_1, j \in P_0 \) where \( w_i < w_j \) to be in our basis, and let \( x_k = 1 - \frac{\eta'}{w_{j^*} - w_{i^*}} \), the objective of (28) would be

\[
 \sum_{k \in \{x\}} (\frac{w_k}{1+W} - c_k)x_k = \sum_{k \in P_1 \setminus \{i\}} (\frac{w_k}{1+W} - c_k)x_k + (1 - \frac{\eta'}{w_{j^*} - w_{i^*}})(\frac{w_i}{1+W} - c_i) + \frac{\eta'}{w_{j^*} - w_{i^*}}(\frac{w_j}{1+W} - c_j)
\]

\[
 = \sum_{k \in P_1} (\frac{w_k}{1+W} - c_k)x_k + \eta'\left(\frac{1}{1+W} - \frac{c_j - c_i}{w_{j^*} - w_{i^*}}\right)
\]
Therefore, to maximize the objective, we should choose \((i^*, j^*) \in \arg\min_{i \in I, j \in J} c_i - c_i \frac{w_i}{w_i - w_j} \).

In summary, the optimal basic solution \(\hat{x}\) to Problem (28) obtained at the end of the dual simplex method must take the following form:

\[
\hat{x}_k = \begin{cases} 
1 - \frac{\eta'}{\eta' - w_{i^*} - w_{j^*}} & k = i^* \\
\frac{w_{j^*} - w_{i^*}}{\eta' - w_{i^*}} & k = j^*. 
\end{cases}
\]

and \(\hat{x}_k = 1\) for any \(k \in P_1 \setminus \{i^*, j^*\}\), \(\hat{x}_k = 1\) for any \(k \in P_0 \setminus \{j^*\}\). This is the desired form given in Equation (26).

Part 2: \(\hat{x}\) is also an optimal basic solution to Problem \((\text{kp-relax}(W + \eta', K))\). To see this, we again consider an equivalent form of Problem \((\text{kp-relax}(W + \eta', K))\) stated below:

\[
\max_{x \in [0, 1]^n, s_1, s_2 \geq 0} \sum_{i \in [n]} \left( \frac{w_i}{1 + W + \eta'} - c_i \right) x_i
\]

subject to

\[
\sum_{i \in [n]} w_i x_i + s_1 = W + \eta'
\]

\[
\sum_{i \in [n]} x_i + s_2 = K
\]

where we introduced slack variables \(s_1, s_2\) to turn inequality constraints into equalities. Note that when we transition from Problem (28) to \((\text{kp-relax}(W + \eta', K))\), we only change the utility of each item. Hence, the feasibility condition still hold. It suffices to check whether the optimality condition hold for each variable.

Applying the same argument as in the proof of Lemma 8, we have that the reduced cost associated with the non-basic variables \(\{x_k : k \neq i^*, j^*\}\) and \(s_2\) are all independent of \(W\). That is, changing the utility of each item \(u_k(W)\) to \(u_k(W + \eta')\) does not change the signs of the reduced costs of \(\{x_k : k \neq i^*, j^*\}\) and \(s_2\), so the optimality condition related to these variables remain unaffected. Since the capacity constraint is binding for \((\text{kp-relax}(W + \eta', K))\), by Lemma 11, we also have that \(c_{j^*} - c_{i^*} \frac{w_{j^*} - w_{i^*}}{\eta' - w_{i^*}} \leq 1 + (W + \eta') - \frac{1}{1 + W + \eta'}\). The reduced cost \(\overline{c}_{s_1}\) associated with slack variable \(s_1\) can be computed to be \(\overline{c}_{s_1} = \frac{c_{j^*} - c_{i^*}}{w_{j^*} - w_{i^*}} - \frac{1}{1 + W + \eta'}\). (See the definition of the reduced costs in Equation (39).\(^{21}\)) Hence, \(\overline{c}_{s_1}\) remains non-positive. Since none of the signs of the reduced costs changed, the optimality condition continues to hold. Considering the fact that both feasibility and optimality conditions hold for solution \(\hat{x}\), it is indeed an optimal basic solution to Problem \((\text{kp-relax}(W + \eta, K))\), and it takes the desired form. \(\blacksquare\)

### F.6.1. Proof of Lemma 9

Recall from Lemma 1 in Caprara et al. (2000) that an optimal basic solution \(x^*\) has at most two fractional variables. Hence, it suffices to show that we cannot have an optimal basic solution \(x^*\) that has only one fractional component.

Suppose by contradiction that \(x^*\) is an optimal basic solution to Problem (28) that has one fractional component \(i\), which implies that the cardinality constraint \(\sum_{i \in [n]} x^*_i \leq K\) is not tight. That is, the slack variable \(s_2 > 0\). In this case, \(x_i\) and \(s_2\) are the two basic variables that take fractional values. Let \(P_1\) denote the indices of variables with value 1, and let \(m = |P_1| \leq K - 1\). Now, suppose that we decrease the capacity limit

\(^{21}\) Here, we used \(B = \{i^*, j^*\}\) as our basis, which gives \(c_B = [u_{i^*}(W + \eta'), u_{j^*}(W + \eta')]^T\) and \(B = \begin{bmatrix} w_{i^*} & w_{j^*} \end{bmatrix}^T\).
Let $x$ be an optimal basic solution to Problem (29). Observe that Problem (29) only differs from Problem (28) in the capacity limit. We note that if we keep $x_i$ and $s_2$ as our basic variables in a basic solution to Problem (29), the optimality condition does not change (since the utility of the items are not changed). Then, of we solve for $x_i$ and $s_2$ using the capacity/cardinality constraints in (29), we have the linear system
\[
\sum_{j \in P_1} w_j + w_i x_i = W' \quad \text{and} \quad \sum_{j \in P_1} x_j + x_i + s_2 = K
\]
which gives $x_i = 0, s_2 = K - m$. Here, the feasibility condition holds because $x_i \in [0, 1]$ and $s_2 \geq 0$. Since both optimality and feasibility conditions hold, the solution $x$ where $x_i = 0$ and $x_k = x'_k$ for all $k \neq i$ is an optimal solution to Problem (29).

We will now derive a contradiction by dividing our analysis into two cases: (i) if $P_1$ contains the $m$ items with the highest utility-to-weight ratios, we cannot have $W \geq w(H_K)$, which violates our assumption in the lemma; (ii) if $P_1$ contains an item that is not among the items with top $m$ utility-to-weight ratios, the solution $x$ cannot be an optimal solution to Problem (29). We now discuss the two possibilities:

- **Case 1**: $P_1$ contains the $m$ items with the highest utility-to-weight ratios. In this case, we have that $W' = \sum_{j \in P_1} w_j = w(H_m) < w(H_K)$. By Lemma 5, we must have $i$ is the item with the $(m+1)$th highest utility-to-weight ratio. (Recall that the order of utility-to-weight ratios only depend on $w_i$ and $c_i$, and does not depend on $W$.) However, this implies that $W + \eta' \leq w(H_K)$, which leads to a contradiction.

- **Case 2**: There exists an item $j \in P_1$ and another item $k \in [n] \setminus P_1$ such that the utility-to-weight ratio of item $k$ is higher than that of item $j$, i.e. $\frac{w_k}{w_j} > \frac{w_j}{w_j}$. We will show, in this case, that $x$ derived in the previous paragraph cannot be optimal for Problem to Problem (29). In particular, we will find another solution $x'$ that yields a higher objective than $x$ in Problem (29). Let us define a new solution $x'$ as follows:
  - if $w_k \geq w_j$, we define $x'$ as
    \[
x'_i = \begin{cases} 
    1 & \text{if } i \in P_1 \text{ and } i \neq j \\
    \frac{w_j}{w_k} & \text{if } i = k \\
    0 & \text{otherwise}
    \end{cases}
    \]
    The solution $x'$ is feasible for Problem (29) since $\sum_{i \in [n]} w_i x'_i = \sum_{j \in P_1} w_j = W'$ and $\sum_{i \in [n]} x'_i = (m - 1) + \frac{w_j}{w_k} < K$. It yields a higher objective value than $x$ in Problem (29), since $\frac{w_j}{w_k} u_k(W) > u_j(W)$.
  - if $w_k < w_j$, we define $x'$ as:
    \[
x'_i = \begin{cases} 
    1 & \text{if } i \in P_1 \setminus \{j\} \text{ or } i = k \\
    \frac{w_j - w_k}{w_j} & \text{if } i = j \\
    0 & \text{otherwise}
    \end{cases}
    \]
This solution $x'$ is feasible for Problem (29) since $\sum_{i\in[n]} w_i x'_i = \sum_{j\in P_k} w_j = W'$ and $\sum_{i\in[n]} x'_i = m + \frac{w_j - w_k}{w_j} < K$. It again yields a higher objective value than $x$ in Problem (29), since $u_k(W) + \frac{w_j - w_k}{w_j} u_j(W) > u_j(W)$.

To summarize, we can always find a feasible solution $x'$ to Problem (29) that gives a higher objective value than $x$; hence, $x$ cannot be optimal for Problem (29). This leads to a contradiction in Case 2.

By proof of contradiction, we thus have that the desired result.

F.6.2. Proof of Lemma 10

To show Lemma 10, we can employ the same proof ideas from Lemma 9. Let $x$ be the optimal solution to Problem (28). Suppose by contradiction that at capacity level $W \geq w(H_K)$, the capacity constraint is binding, but the cardinality constraint is non-binding. That is, $\sum_{i\in[n]} w_i x_i^* = W + \eta'$ and $\sum_{i\in[n]} x_i^* < K$. Since the capacity constraint is binding, we must have $s_2 > 0$ as one of the basic variables in the optimal solution. Since the capacity constraint is binding, we also have $s_1 = 0$. Then, our optimal basic solution $x^*$ can take one of the following two forms: (i) $x^*$ has one fractional component (ii) $x^*$ is an integer solution with $\sum_{i=1}^n x_i^* = m \leq K - 1$. We have showed, in Lemma 9, that the first form is not possible. We can show that the second form cannot be an optimal solution to Problem (28), using the same argument as in the proof of Lemma 9. That is, if the optimal solution contains the $m$ items with highest utility-to-weight ratios, we would have $W + \eta' = w(H_m) < w(H_K)$ (which is a contradiction); otherwise, we can find another solution $x'$ which yields higher utilities (which contradicts the optimality of $x^*$). Thus, using contradiction, we show the desired result.

F.7. Proof of Lemma 8

Suppose that at capacity level $\tilde{W} > w(H_K)$, an optimal basic solution $x^*(\tilde{W})$ to (KP-RELAX(\tilde{W}, K)) has profile $P(\tilde{W}) = \{P_1 \setminus \{i\}, (i,j), P_0 \setminus \{j\}\}$. Fix $\tilde{W} \in [w(P_1), w(P_1) - w_i + w_j]$ such that $\frac{w_i - w_j}{w_j - w_i} \leq \frac{1}{\tilde{W}}$. For simplicity of notation, we let $x^* = x^*(\tilde{W})$ and $x = x^*(\tilde{W})$. As shown in Caprara et al. (2000), in the optimal basic solution of a relaxed knapsack problem KP-RELAX(W, K), we have at most two basic variables. Here, at capacity level $\tilde{W}$, $x_i^*$ and $x_j^*$ are the two fractional variables in the optimal basic solution $x^*$. In the following proof, we will show that if we change the capacity level from $\tilde{W}$ to $\hat{W}$, and $x_i$ and $x_j$ as our two basic variables will continue to be the two basic variables in the optimal solution to Problem (KP-RELAX(\hat{W}, K)).

To show that, we consider the optimality condition and feasibility condition defined in Appendix J, and show that both of them hold when we have $\{i,j\}$ as our basis. This then allows us to solve for $x = x^*(\hat{W})$ using a linear system defined by capacity and cardinality constraints.

**Optimality condition.** Since $x^*$ is a nondegenerate, optimal basic solution to (KP-RELAX(\hat{W}, K)), we know from Appendix J that it satisfies the optimality condition (i.e., all reduced costs are non-positive). If we change the capacity level to $\hat{W}$ and consider (KP-RELAX(\hat{W}, K)), with $x_i$ and $x_j$ as the two basic variables, we claim that the optimality conditions still hold. To see that, we consider (KP-RELAX(\hat{W}, K)) in its equivalent form (KP-RELAX-EQ(\hat{W}, K)). Since $x_i^*$ and $x_j^*$ are the two basic variables in the optimal solution for (KP-RELAX-EQ(\hat{W}, K)) (which share the same optimal solution with (KP-RELAX(\hat{W}, K))), we know that for any non-basic variable $x_k^*$, the reduced cost $\overline{c}_k \leq 0$ if $x_k^* = 0$ and $\overline{c}_k \geq 0$ if $x_k^* = 1$ (this is because if $x_k^*$ is at its lower bound 0, when we increase $x_k^*$ our objective increases by the reduced cost; if $x_k^*$
is at its upper bound 1, when we decrease \( x_k^* \), our objective decreases by the reduced cost. The non-basic variables \( s_1, s_2 \), also have non-positive reduced costs. Let us first focus on any non-basic variables with \( x_k^* = 0 \).

The corresponding reduced cost of this variable is a follows: (see the definition of reduced costs in Equation (39),\(^{22}\))

\[
\bar{c}_k = \left( \frac{w_k}{1+W} - c_k \right) + \left( \frac{w_i}{1+W} - c_i \right) \frac{w_k - w_j}{w_j - w_i} - \left( \frac{w_j}{1+W} - c_j \right) \frac{w_k - w_i}{w_j - w_i} \leq 0
\]

\[
\iff (w_j - w_i)[w_k - c_k(1+\hat{W})] + (w_k - w_j)[w_i - c_i(1+\hat{W})] - (w_k - w_i)[w_j - c_j(1+\hat{W})] \leq 0
\]

\[
\iff (1+\hat{W})[(w_i - w_j)c_k + (w_j - w_k)c_i + (w_k - w_i)c_j] \leq 0
\]

Note that the sign of the reduced cost of \( x_k^* \) is in fact independent of the capacity level \( \hat{W} \). That is, if we replace \( \hat{W} \) with \( \hat{W} \) and consider \((\text{KP-relax-eq}(\hat{W}, K))\) with \( x_i \) and \( x_j \) as the two basic variables, the sign of the reduced cost of \( x_k \) is the same as the sign of the reduced cost of \( x_k^* \). A similar argument applies to non-basic variables \( x_k^* \) that takes value 1. We also have that the reduced cost of \( s_1 \) is non-positive by our assumption:

\[
\bar{c}_{s_1} = \frac{c_j - c_i}{w_j - w_i} - \frac{1}{1+\hat{W}} \leq 0.
\]

Finally, we compute the reduced cost of \( s_2 \):

\[
\bar{c}_{s_2} = \frac{1}{w_j - w_i}[c_i w_j - c_j w_i],
\]

which is again independent of the capacity level, so replacing \( \hat{W} \) with \( \hat{W} \) does not change the sign of the reduced cost of \( s_2 \). Overall, we have showed that optimality conditions are unaffected when we move from \((\text{KP-relax}(\hat{W}, K))\) to \((\text{KP-relax}(\hat{W}, K))\).

**Feasibility condition.** It suffices to check if the feasibility condition still holds for \((\text{KP-relax}(\hat{W}, K))\) with \( x_i \) and \( x_j \) as the two basic variables. To check that, we solve the following linear system:

\[
w_i x_i + w_j x_j = \hat{W} - \sum_{k \in P_1 \setminus \{i\}} w_k
\]

\[
x_i + x_j = 1,
\]

where the first linear equation ensures that the capacity constraint is binding at solution \( x \), and the second linear equation ensures that the cardinality constraint is binding at solution \( x \). The aforementioned linear system has the following solution:

\[
x_i = \frac{\hat{W} - \sum_{k \in P_1} w_k + w_i - w_j}{w_i - w_j}
\]

and

\[
x_j = \frac{\hat{W} - \sum_{k \in P_1} w_k}{w_j - w_i}.
\]

(30)

It is then easy to verify that for any \( \hat{W} \in [\sum_{k \in P_1} w_k, \sum_{k \in P_1} w_k - w_i + w_j] \), we have \( x_i, x_j \), computed above falls into the interval \([0, 1]\). That is, the basic solution with \( x_i \) and \( x_j \) as the two basic variables is always feasible for \( \hat{W} \in [\sum_{k \in P_1} w_k, \sum_{k \in P_1} w_k - w_i + w_j] \). Hence, the feasibility condition also holds.

To summarize, when we transition from Problem \((\text{KP-relax}(\hat{W}, K))\) to Problem \((\text{KP-relax}(\hat{W}, K))\) and keep \( x_i, x_j \) as our two basic variables, both the optimality and feasibility conditions continue to hold. Hence, the optimal solution to Problem \((\text{KP-relax}(\hat{W}, K))\) satisfies that \( x_k = x_k^* \) for any \( k \neq i, j \), while \( x_i \) and \( x_j \) takes the form in Equation (30). We thus show the desired result. \( \blacksquare \)

\(^{22}\) Here, we used \( B = \{i, j\} \) as our basis, which gives \( c_B = [u_i(\hat{W}), u_j(\hat{W})]^\top \) and \( B = \begin{bmatrix} w_i & w_j \\ 1 & 1 \end{bmatrix} \).
F.8. Other Lemmas

Lemma 11. Suppose that at capacity level $W$, the capacity constraint is binding and the relaxed knapsack problem \((\text{kp-relax}(W, K))\) has a degenerate integer optimal basic solution $x^*$, with profile $P(W) = \{P_1, (0,0), P_0\}$. Then, if there exist $i \in P_1$, $j \in P_0$ such that $w_i < w_j$, and $(i^*, j^*) = \arg\min_{i \in P_1, j \in P_0} \frac{c_j - c_i}{w_j - w_i}$, for $0 < \eta \leq w_j - w_i$, the capacity constraint is nonbinding for Problem \((\text{kp-relax}(W + \eta, K))\) if

$$\frac{c_{j^*} - c_{i^*}}{w_{j^*} - w_{i^*}} > \frac{1}{1 + (W + \eta)}.$$  

We remark that the proof for Lemma 11 is very similar to the proof for Lemma 4. In fact, Lemma 4 can be considered as a special case of Lemma 11, if we take $W = W^{(\ell - 1)}_\text{next}, W + \eta = W^{(\ell)}_\text{next}, P_1 = P^{(\ell - 1)}_1, P_0 = P^{(\ell - 1)}_0$. We include the proof for Lemma 11 here for completeness.

Proof of Lemma 11. We first note that for any $i,j \in [n]$, we have the following equivalence:

$$\frac{c_j - c_i}{w_j - w_i} > \frac{1}{1 + W + \eta} \iff \frac{w_i}{1 + W + \eta} - c_i > \frac{w_j}{1 + W + \eta} - c_j \iff u_i(W + \eta) > u_j(W + \eta).$$

Suppose $\frac{c_{j^*} - c_{i^*}}{w_{j^*} - w_{i^*}} > \frac{1}{1 + W + \eta}$, where $(i^*, j^*) \in \arg\min_{i \in P_1, j \in P_0} \frac{c_j - c_i}{w_j - w_i}$. We must then have $u_i(W + \eta) > u_j(W + \eta)$ for any $i \in P_1, j \in P_0$ with $w_i < w_j$. That is, for Problem \((\text{kp-relax}(W + \eta, K))\), the $K$ items with the highest utilities $u_i(W + \eta)$ are all included in $P_1$. Hence, the optimal solution to Problem \((\text{kp-relax}(W + \eta, K))\) is $S^* = \{ i \in P_1 : u_i(W + \eta) \geq 0 \}$. The set $S^*$ is also a feasible solution to Problem \((\text{kp-relax}(W + \eta, K))\). This is because the capacity constraint is binding for Problem \((\text{kp-relax}(W, K))\), and hence $w(S^*) \leq w(P_1) = W < W + \eta$. This chain of inequality shows that the capacity constraint is non-binding for Problem \((\text{kp-relax}(W + \eta, K))\).  

Appendix G: PTAS for Problem \((\text{sub-dual}(z, K))\)

In this section, we briefly discuss how we can design a PTAS for Problem \((\text{sub-dual}(z, K))\) with the help of our 1/2-approx. algorithm for Problem \((\text{sub-dual}(z, K))\). The design of the PTAS has a similar flavor as the PTAS presented in Caprara et al. (2000).

Before applying the PTAS, we first pre-partition the interval $[0, \infty)$ into $O(n^2)$ sub-intervals, such that on each sub-interval $I$, the ordering of utilities of items do not change. Note that when the utilities take the form $u_i = \frac{w_i}{1 + W} - c_i$, the ordering of utilities only changes at values of $W$ such that

$$\frac{w_i}{1 + W} - c_i = \frac{w_j}{1 + W} - c_j \iff W = \frac{w_i - w_j}{c_i - c_j} - 1$$

for some $i \neq j$. Hence, there are at most $O(n^2)$ such values of $W$. Next, we focus on such a sub-interval $I$ and we design a PTAS for the following problem: $\max_{W \in I} \text{kp}(W, K)$. Our PTAS returns an assortment $S_I$ with $|S_I| \leq K$ such that $\text{ms-cost}(S_I) \geq (1 - \epsilon) \max_{W \in I} \text{kp}(W, K)$.

The details of the PTAS are outlined in Algorithm 4. The algorithm maintains a collection of assortments $C_I$ and returns the best of this collection as $S_I$. This collection $C_I$ contains all the assortments with small size, i.e., size less than $\ell$, where $\ell = \min\{\lceil \frac{1}{2} - 2 \rceil, K\}$. (See step 2 of Algorithm 4.) The collection also contains feasible assortments with size greater than $\ell$. To do so, the algorithm considers all feasible assortments $L$ with size $\ell$ and total weight less than $W_{\text{max}}$, i.e., $w(L) \leq W_{\text{max}}$. The algorithm then appends a subset of
Algorithm 4 PTAS for $\max_{W \in I} \text{KP}(W,K)$

Input: weights $w = (w_1, \ldots, w_n)$, costs $c = (c_1, \ldots, c_n)$, cardinality upper bound $K$, interval $I = [W_{\min}, W_{\max}] \subset [0, \infty)$, accuracy parameter $\epsilon \in (0, 1)$, an 1/2-approx. algorithm $A$ for $\max_{W \in I} \text{KP}(W,K)$.

Output: assortment $S_I$.

1. Let $\ell = \min(\lfloor \frac{1}{\epsilon} - 2 \rfloor, K)$. Initialize the collection of assortments $C_I = \emptyset$.
2. For each $L \subset [n]$ such that $|L| \leq \ell - 1$ and $w(L) \leq W_{\max}$, add $L$ to $C_I$.
3. For each $L \subset [n]$ such that $|L| = \ell$ and $w(L) \leq W_{\max}$:
   - Let $\text{sm}(L)$ be the set of items that are not in $L$, and have smaller utilities than the items in set $L$:
     \[
     \text{sm}(L) = \left\{ i \in [n] \setminus L : u_i(W) \leq \min_{j \in L} \{ u_j(W) \} \text{ for any } W \in I \right\}.
     \]
     Note that $\text{sm}(L)$ is well-defined since the ordering of utilities does not change on $I$.
   - Consider the sub-knapsack problem defined on $\text{sm}(L)$:
     \[
     \text{KP-sm}(W,K,L) = \max_{S \subseteq \text{sm}(L)} \sum_{i \in S} u_i(W) \quad \text{s.t.} \quad \sum_{i \in S} w_i \leq W - w(L) \quad \text{and} \quad |S| \leq K - \ell,
     \]
     Apply the 1/2-approx. algorithm $A$ to solving $\max_{W \in I} \text{KP-sm}(W,K,L)$, which returns a collection of sets $C_{1/2}(L)$ that contains all the sets that could be an integer 1/2-approx. solution to the problem $\max_{W \in I} \text{KP-sm}(W,K,L)$.
   - For each $S$ in $C_{1/2}(L)$, add $L \cup S$ to $C_I$.
4. Return the assortment $S_I = \arg \max_{S \in C_I} \text{MS-Cost}(S)$.

items with small utilities to set $L$. For any given subset $L \subset [n]$, let us define the set of items, denoted by $\text{sm}(L)$, that are not in $L$, and have smaller utilities than any item in $L$ (“sm()” stands for small.). That is,

\[
\text{sm}(L) = \left\{ i \in [n] \setminus L : u_i(W) \leq \min_{j \in L} \{ u_j(W) \} \text{ for any } W \in I \right\}.
\]

Note that set $\text{sm}(L)$ is well-defined as the order of utilities does not change when $W \in I$, and hence if $\frac{w_i}{1 + \epsilon} - c_i \leq \min_{j \in L} \left\{ \frac{w_j}{1 + \epsilon} - c_j \right\}$ for some $W \in I$, the same holds for any other $W' \in I$. The algorithm would like to choose some subsets of set $\text{sm}(L)$, denoted by $C_{1/2}(L)$, and append them to set $L$. To do that, the algorithm considers the following sub-knapsack problem defined on $\text{sm}(L)$:

\[
\text{KP-sm}(W,K,L) = \max_{S \subseteq \text{sm}(L)} \sum_{i \in S} u_i(W) \quad \text{s.t.} \quad \sum_{i \in S} w_i \leq W - w(L) \quad \text{and} \quad |S| \leq K - \ell
\]

and applies the 1/2-approx. algorithm to solving $\max_{W \in I} \text{KP-sm}(W,K,L)$. Note that instead of having the 1/2-approx. algorithm return the 1/2-approx. solution for $\max_{W \in I} \text{KP-sm}(W,K,L)$ at the termination step, we instead consider the collection it maintains at the end of the algorithm, which we denote as $C_{1/2}(L)$. This is the collection that contains all the sets that can be an integer 1/2-approx. solution to $\max_{W \in I} \text{KP-sm}(W,K,L)$.

The following proposition states that the assortment $S_I$ returned by Algorithm 4 is indeed near-optimal, and runs in time polynomial in $n$ and $K$. 
**Proposition 3 (Near-optimality of Algorithm 4).** Consider any interval $I$ such that on this interval the ordering of utilities of items do not change. Then, for any $z \geq 0$, set $S_I$, returned by Algorithm 4, has the following property

$$\text{MS-COST}(S_I, z) \geq (1 - \epsilon) \max_{W \in I} \text{KP}(W, K)$$

In addition, the overall complexity of Algorithm 4 is in the order of $O((n^{1/\epsilon})(\log n + K))$.

**Proof of Proposition 3** The proof that shows Algorithm 4 returns a $(1 - \epsilon)$-approx. solution uses the same idea as the proof for Proposition 3 in Caprara et al. (2000).

Let us first fix $W \in I$, and let $S^*$ be the optimal solution to the knapsack problem $\text{KP}(W, K)$. We will show that the set $S_I$ returned by Algorithm 4 satisfies

$$\text{MS-COST}(S_I, z) \geq (1 - \epsilon)\text{KP}(W, K)$$

regardless of our choice of $W$. Clearly, if $|S^*|$ contains less than $\ell$ items, then $S^*$ must have been added to $C_I$ in Step 2 of Algorithm 4. Hence,

$$\text{MS-COST}(S_I, z) \geq \text{MS-COST}(S^*, z) = \text{KP}(W, K).$$

Now, if $|S^*|$ contains more than $\ell$ items, we can partition $S^*$ into $S^* = L^* \cup M^*$, where $L^*$ contains the $\ell$ items in $S^*$ with the highest utilities and $M^*$ include the rest of the items. Note that we must have $M^* \subseteq \text{sm}(L^*)$, and that $M^*$ is the optimal solution to the sub-knapsack problem ($\text{KP-sm}(W, K, L^*)$). Hence, we must have

$$\text{KP}(W, K) = \sum_{i \in L^* \cup M^*} u_i(W) = \sum_{i \in L^*} u_i(W) + \text{KP-sm}(W, K, L^*).$$

In one of the iterations of Step 3 of Algorithm 4, the set $L$ we consider is $L^*$, and we apply our 1/2-approx. algorithm to solving Problem ($\text{KP-sm}(W, K, L^*)$). By Claim 2 in the proof of Theorem 5, we have the the collection of sets $C_{1/2}(L^*)$ contains an integer, 1/2-approx. feasible solution for the sub-knapsack problem ($\text{KP-sm}(W, K, L^*)$). That is, there exists $M \in C_{1/2}(L^*)$ such that

$$\sum_{i \in M} u_i(W) \geq \frac{1}{2} \text{KP-sm}(W, K, L^*) = \frac{1}{2} \sum_{i \in M} u_i(W). \quad (32)$$

In Step 3 of Algorithm 4, we have added $L^* \cup M$ to the collection $C_I$.

We now show the key inequality below:

$$\sum_{i \in L^*} u_i(W) + \sum_{i \in M} u_i(W) \geq \frac{\ell + 1}{\ell + 2} \text{KP}(W, K). \quad (33)$$

To show that, we consider the following two cases:

1. **The total utility of items in $L^*$ is big enough:** $\sum_{i \in L^*} u_i(W) \geq \frac{\ell}{\ell + 2} \text{KP}(W, K)$. Intuitively, this implies that approximating $M^*$ with $M$ does not hurt our total utility by a lot. We have

$$\sum_{i \in L^*} u_i(W) + \sum_{i \in M} u_i(W) \geq \sum_{i \in L^*} u_i(W) + \frac{1}{2} \sum_{i \in M} u_i(W)$$

by (32)

$$= \sum_{i \in L^*} u_i(W) + \frac{1}{2} \left( \text{KP}(W, K) - \sum_{i \in L^*} u_i(W) \right)$$

$$= \frac{1}{2} \left( \text{KP}(W, K) + \sum_{i \in L^*} u_i(W) \right)$$

$$\geq \frac{\ell + 1}{\ell + 2} \text{KP}(W, K)$$

since $\sum_{i \in L^*} u_i(W) \geq \frac{\ell}{\ell + 2} \text{KP}(W, K)$.  

(2) The items in $L^*$ have small utilities: $\sum_{i \in L^*} u_i(W) < \frac{1}{\ell+2} \text{KP}(W, K)$. In this case, the smallest utility of item in $L^*$ must be bounded by $\frac{1}{\ell+2} \text{KP}(W, K)$. Hence, all items in $\text{sm}(L^*)$ have utilities less than or equal to $\frac{1}{\ell+2} \text{KP}(W, K)$. If the optimal solution to the LP relaxation of $(\text{KP-sm}(W, K, L^*))$ is integer, then by Lemma 1 and design of our 1/2-approx. algorithm, it returns an optimal solution to $(\text{KP-sm}(W, K, L^*))$. Otherwise, if the optimal solution to the LP relaxation of $(\text{KP-sm}(W, K, L^*))$ is fractional, by Lemma 1, design of our 1/2-approx. algorithm as well as Lemma 1 in Caprara et al. (2000), we know that

$$\text{KP-sm}(W, K, L^*) - \sum_{i \in M} u_i(W) \leq \max_{i \in \text{sm}(L^*)} u_i(W) < \frac{1}{\ell+2} \text{KP}(W, K).$$

This then gives

$$\sum_{i \in L^*} u_i(W) + \sum_{i \in M} u_i(W) \geq \sum_{i \in L^*} u_i(W) + \text{KP-sm}(W, K, L^*) - \frac{1}{\ell+2} \text{KP}(W, K) = \frac{\ell+1}{\ell+2} \text{KP}(W, K).$$

The two cases above establish the inequality in (33). Now, plugging in $\ell = \min\{\lceil \frac{1}{\epsilon} - 2 \rceil, K\}$ gives the desired approximation factor $(1 - \epsilon)$.

In terms of the runtime of Algorithm 4, it is clear that in Step 2, the algorithm considers $O(n^{\ell - 1})$ subsets. In Step 3, the algorithm iterates over $O(n^{\ell})$ subsets, while for each subset, it executes the 1/2-approx. algorithm (Algorithm 1) once to solve the sub-knapsack problem defined on $\text{sm}(L)$, which takes $O(n^2(\log n + K))$ time. So the overall complexity of Algorithm is $O(n^{\ell + 2}(\log n + K)) = O(n^{1/\epsilon + 1}(\log n + K))$.

Recall that we have pre-partitioned $[0, \infty)$ to $O(n^2)$ sub-intervals. Now that we have a PTAS for $\max_{W \in I} \text{KP}(W, K)$ for each sub-interval $I$, we can apply Algorithm 4 separately to each sub-interval $I$ and out of the returned assortments, we pick the one with the highest cost-adjusted market share. This assortment would then be a $(1 - \epsilon)$-approximate solution to Problem (SUB-DUAL($z, K$)). The overall running time of our PTAS is $O(n^{1/\epsilon + 2}(\log n + K))$, since we need to apply Algorithm 4 to $O(n^2)$ sub-intervals. A similar argument is used in the design of our FPTAS (see Section 5.3).

**Appendix H: Proof of Theorem 6**

The proof consists of two segments. In the first segment, we show that

$$\text{ms-cost}(S_I, z) \geq (1 - \epsilon) \max_{W \in I} \text{KP}(W, K),$$

where $S_I$ is the solution returned by Algorithm 2. In the second segment, we bound the running time of the algorithm.

**Segment 1.** Suppose the maximum $\max_{W \in I} \text{KP}(W, K)$ is achieved at $W^* \in I$ with assortment $S^*$. We divide our analysis into two cases. In the first case, we have $S^* \in C_I$, and in the second case, which is the more challenging case, $S^* \notin C_I$. In the second case, we find a set $S' \in C_I$, which is close to $S^*$ in the sense that the sum of scaled utility under $S^*$ and $S'$ are the same. We then show that the total weight under $S'$ is less than that of $S^*$. This and our re-scaling scheme allows us to show $\text{ms-cost}(S', z) \geq (1 - \epsilon) \max_{W \in I} \text{KP}(W, K)$, which is the desired result.

Note that by definition of the eligible set $E_I$ (Equation 6), items not in $E_I$ either cannot fit into the knapsack of size $W_{\min}$ or have a negative utility. Hence, any item $j \notin E_I$ will not be in the optimal assortment.
$S^\ast$. That is, $S^\ast \subseteq E_1$. In Step 3 of Algorithm 2, we also ensure that we do not include any of the items $j \notin E_1$ in any of the assortments $\text{set}_i(.)$ for all $i \in [n]$. In the following, we can assume that any items $j \in \text{set}_i(.)$, $i \in [n]$ have a non-negative utility and can fit into the knapsack of size $W_{\min}$. We now proceed to discuss the two cases.

1. $S^\ast \in \mathcal{C}_1$. In this case, we have the following:

$$\text{MS-COST}(S_1, \mathbf{z}) \geq \text{MS-COST}(S^\ast, \mathbf{z}) = \sum_{i \in S^\ast} \frac{w_i}{1 + \sum_{j \in S^\ast} w_i} - \sum_{i \in S^\ast} c_i \geq \sum_{i \in S^\ast} \frac{w_i}{1 + W^*} - \sum_{i \in S^\ast} c_i = \text{KP}(W^*, K) = \max_{W \in \mathcal{I}} \text{KP}(W, K),$$

where the first inequality follows from Step 5 of Algorithm 2 and the assumption that $S^\ast \in \mathcal{C}_1$. The second inequality follows from the fact that $S^\ast$ is a feasible solution to Problem $\text{KP}(W^*, K)$, which gives $W^* \geq \sum_{i \in S^\ast} w_i$.

2. $S^\ast \notin \mathcal{C}_1$. Suppose there are $b^\ast$ items in $S^\ast$ and let $a^* = \sum_{i \in S^\ast} w_i$, where $w_i = \left\lceil \frac{u_i(W_{\min}K)}{U\epsilon} \right\rceil$ is the re-scaled utility of item $i$. Since $\sum_{i \in S^\ast} w_i(W_{\min}) \leq KU$, after re-scaling, we have $a^* = \sum_{i \in S^\ast} w_i \leq KU \cdot \frac{K}{U\epsilon} + K \leq \lceil \frac{K^2}{\epsilon} \rceil + K = a_{\max}$. Then, because $a^* \leq a_{\max}$ and $b^* \leq K$, we must have

$$\text{MIN-WT}_n(a^*, b^*) \leq \sum_{i \in S^\ast} w_i \leq W^* \leq W_{\max},$$

where the first inequality holds because $\text{MIN-WT}_n(a^*, b^*)$ is the minimum weight that one would need, in order to generate total utility $a^*$ with $b^*$ items out of the $n$ items.

Set $S' \in \mathcal{C}_1$ and its properties. Since $S^\ast \notin \mathcal{C}_1$, there exists set $S' \in \mathcal{C}_1$ such that $S' = \text{set}_n(a^*, b^*)$. Observe that by definition of $\text{set}_n(a^*, b^*)$, sets $S^\ast$ and $S'$ have the same total scaled utilities. That is,

$$\sum_{i \in S'} \left(\frac{w_i}{1 + W_{\min}} - c_i\right)KU \epsilon = \sum_{i \in S^\ast} \left(\frac{w_i}{1 + W_{\min}} - c_i\right)KU \epsilon = a^*$$

$$\Rightarrow \sum_{i \in S'} \left(\frac{w_i}{1 + W_{\min}} - c_i\right)KU \epsilon + K \geq \sum_{i \in S^\ast} \left(\frac{w_i}{1 + W_{\min}} - c_i\right)KU \epsilon \Rightarrow \sum_{i \in S'} (\frac{w_i}{1 + W_{\min}} - c_i) + U \epsilon \geq \sum_{i \in S^\ast} (\frac{w_i}{1 + W_{\min}} - c_i),$$

where the last inequality is obtained by multiplying both sides of the second inequality by $U\epsilon/K$. In addition, the total weights of $S'$ is no more than the total weights in $S^\ast$. That is,

$$\sum_{i \in S'} w_i \leq \sum_{i \in S^\ast} w_i \leq W^* \leq W_{\max}.$$  

This is because the $S' = \text{set}_n(a^*, b^*)$ is the set with the minimum weight that generate total utility $a^*$ with $b^*$ items out of the $n$ items. Note that Equation (35) implies that $S'$ is also a feasible solution for Problem $(\text{KP}(W^*, K))$. Since $\sum_{i \in S'} w_i \leq \sum_{i \in S^\ast} w_i$ by Equation (35), and $W_{\min} \leq W^*$, we must have

$$\sum_{i \in S'} w_i (\frac{1}{1 + W^*} - \frac{1}{1 + W_{\min}}) = \frac{W_{\min} - W^*}{(1 + W^*)(1 + W_{\min})} \sum_{i \in S'} w_i \geq \frac{W_{\min} - W^*}{(1 + W^*)(1 + W_{\min})} \sum_{i \in S^\ast} w_i = \sum_{i \in S^\ast} w_i (\frac{1}{1 + W^*} - \frac{1}{1 + W_{\min}})$$

(36)
Adding Equations (34) and (36) gives
\[
\sum_{i \in S'} \left( \frac{w_i}{1 + W^*} - c_i \right) + U \epsilon \geq \sum_{i \in S'} \left( \frac{w_i}{1 + W^*} - c_i \right) = \max_{W \in \mathcal{I}} \kappa P(W, K).
\]
(37)

Considering the fact that any single item in the eligible set \( E \), defined in Equation (6), satisfies \( w_i \leq W_{\text{min}} \), we must have \( \max_{W \in \mathcal{I}} \kappa P(W, K) \geq U = \max_{u \in E} u(W_{\text{min}}) \). This is because for Problem \((\kappa P(W_{\text{min}}, K))\), the set consisting of a single item with the highest utility would be a feasible solution. This (i.e., \( \max_{W \in \mathcal{I}} \kappa P(W, K) \geq U \)), combined with Equation (37), gives
\[
\sum_{i \in S'} \left( \frac{w_i}{1 + W^*} - c_i \right) + \epsilon \cdot \max_{W \in \mathcal{I}} \kappa P(W, K) \geq \max_{W \in \mathcal{I}} \kappa P(W, K).
\]
(38)

Now, since \( S' \in \mathcal{C}_J \), we have
\[
\text{MS-COST}(S_I, z) \geq \text{MS-COST}(S', z) \geq \sum_{i \in S'} \frac{w_i}{1 + \sum_{j \in S'} w_i} - \sum_{i \in S'} c_i
\]
\[
\geq \sum_{i \in S'} \frac{w_i}{1 + W^*} - \sum_{i \in S'} c_i
\]
\[
\geq (1 - \epsilon) \max_{W \in \mathcal{I}} \kappa P(W, K),
\]
where the first inequality follows from Step 5 of Algorithm 2; the second inequality follows from Equation (35); the third inequality follows from Equation (38).

In summary, we have showed, in both cases, that \( \text{MS-COST}(S_I, z) \geq (1 - \epsilon) \max_{W \in \mathcal{I}} \kappa P(W, K) \), which is the desired result.

**Segment 2.** We now comment on the running time of our FPTAS. We start by analyzing Algorithm 2. Step 1 of Algorithm 2 takes \( O(n) \) since it simply rescales the utilities of each item. In Step 2 and 3, we perform a dynamic programming scheme, which operates on a matrix of size \( O(n \cdot a_{\text{max}} \cdot K) = O(nK([K^2/\epsilon] + K)) = O(nK^3/\epsilon) \). Computing each entry of the matrix \( (\text{MIN-WT}_i(a, b)) \) takes \( O(1) \). For each entry of the matrix, updating its corresponding assortment \( \text{SET}_i(a, b) \) also takes \( O(1) \). Hence the overall complexity of Step 2 and 3 is \( O(nK^3/\epsilon) \). In Step 4, we collect at most \( O(a_{\text{max}} \cdot K) = O(K^3/\epsilon) \) sets \( \text{MIN-WT}_n(a, b) \). In Step 5, finding the set that maximizes \( \text{MS-COST}(S, z) \) again takes \( O(K^3/\epsilon) \). In summary, the overall complexity of Algorithm 2 is \( O(nK^3/\epsilon) \).

**Appendix I:** Modified 1/2-Approx. Algorithm for Problem \((\text{SUB-DUAL}(z, K))\) with Profits

In Section 7.1, we consider the more general setting where the sale of each item \( i \) generates profit \( r_i > 0 \). Recall that we reformulate Problem \((\text{SUB-DUAL}(z, K))\) as follows
\[
\text{SUB-DUAL}(z, K) = \max_{S: |S| \leq K} \left\{ \text{REV}(S) - \sum_{i \in S} c_i(z) \right\}.
\]

Here, we provide the details about the modified 1/2-approx. algorithm for Problem \((\text{SUB-DUAL}(z, K))\). Recall from 7.1 that we first need to pre-partition the interval \([0, \infty)\) into \( O(n^3) \) sub-intervals, such that on each sub-interval \( I \), we have that (i) the ordering of utilities of items do not change; (ii) the ordering of utility-to-weight ratios of items do not change; (iii) for any three items \((i, j, k)\), the sign of the reduced cost of item \( k \) with items \( i, j \) as the basic variables does not change. After the pre-partitioning step, we apply the following
Algorithm 5 Modified 1/2-approx. algorithm for max$_{W \in I}$ KP($W, K$)

**Input:** weights $w = (w_1, \ldots, w_n)$, costs $c = (c_1, \ldots, c_n)$, cardinality upper bound $K$, interval $I = [W_{\min}, W_{\max}] \subset [0, \infty)$.

**Output:** assortment $S_{1/2}$.

1. **Initialization.**
   
   (a) Initialize the collection of assortments $C = \emptyset$. For $j \in [n]$, add $\{j\}$ to $C$.
   
   (b) Rank the items by their utility-to-weight ratios. Let $h_j$ be the index of the item with the $j$th highest utility-to-weight ratio, for $j \in [n]$, and define $H_j = \{h_1, \ldots, h_j\}$, for $j \in [K]$ and $W_m = w(H_K)$.

2. **Interval** $I_{\text{low}} = I \cap [0, W_{\text{th}})$. If $I_{\text{low}}$ is non-empty:
   
   (a) For $j = 1, \ldots, K$, add $H_j$ to $C$.
   
   (b) **Stopping rule.** If $u_{th}(W_m) \geq 0$, go to Step 3; otherwise, go to Step 4 (i.e., the termination step).

3. **Interval** $I_{\text{high}} = I \cap [W_{\text{th}}, \infty)$. If $I_{\text{high}}$ is non-empty:

   (a) **Initialize the profile.**
   
   - If $W_{\text{min}} \leq W_{\text{th}}$, set $P_1 = H_K$, $P_0 = [n] \setminus H_K$, and $W_{\text{next}} = W_m$.
   
   - If $W_{\text{min}} > W_{\text{th}}$, compute an optimal basic solution for Problem (KP-relax($W_{\text{min}}, K$)) with basis $\mathcal{P}(W_{\text{min}}) = \{P_1(W_{\text{min}}), \{i\}, P_0(W_{\text{min}})\}$, where $w_i < w_j$. Add $P_1(W_{\text{min}}) \cup \{i\}$ and $P_1(W_{\text{min}}) \cup \{j\}$ to $C$. Set $P_1 = P_1(W_{\text{min}}) \cup \{j\}$, $P_0 = [n] \setminus P_1$, and $W_{\text{next}} = w(P_1)$.

   (b) **Adaptively partitioning** $I_{\text{high}}$. While there exist $i \in P_1, j \in P_0$ such that $w_i < w_j$:
   
   i. Update indices $i^*, j^*$ as follows:
   
   $$ (i^*, j^*) \leftarrow \arg \max_{i^* \in P_1, j^* \in P_0, w_{i^*} < w_{j^*}} \left[ \frac{u_{j^*}(W_{\text{next}}) - u_{i^*}(W_{\text{next}})}{w_{j^*} - w_{i^*}} \right]. $$

   ii. **Stopping rule.** If $u_{i^*}(W_{\text{next}}) > u_{j^*}(W_{\text{next}})$, go to the termination Step 4.

   iii. **Swapping the two items.** Update $P_1, P_0, W_{\text{next}}$ as follows:
   
   $$ P_1 \leftarrow P_1 \cup \{j^*\} \setminus \{i^*\} \quad \text{and} \quad P_0 \leftarrow P_0 \cup \{i^*\} \setminus \{j^*\} \quad \text{and} \quad W_{\text{next}} \leftarrow W_{\text{next}} - w_{i^*} + w_{j^*}. $$

   Add $P_1$ to $C$.

4. **Termination Step.** Return $S_{1/2} = \arg \max_{S \in C} \text{MS-COST}(S, z)$.

algorithm presented in Algorithm 5 to solving max$_{W \in I}$ KP($W, K$). Note that the utilities in Algorithm 5 now depend on the profits, and are defined as $u_i(W) = \frac{c_i W}{W + 1} - c_i$.

After we apply Algorithm 5 to solving max$_{W \in I}$ KP($W, K$) for each sub-interval $I$ separately, we pick the assortment with the highest cost-adjusted market share out of all returned assortments. This assortment would then be a 1/2-approximate solution to Problem (sub-dual($z, K$)). The overall running time of our 1/2-approx. algorithm is $O(n^5(\log n + K))$, since Algorithm 5 has the same $O(n^2(\log n + K))$ complexity as Algorithm 1, but we need to apply Algorithm 5 to $O(n^3)$ sub-intervals.

**Appendix J: Backgrounds on Linear Programming**

In this section, we provide an brief overview of the key terminologies in linear programming that we used throughout the paper. For an more detailed discussion, see Bertsimas and Tsitsiklis (1997).
In our overview, we consider a linear program in its standard form, i.e.

$$\max_x \ c^\top x$$

s.t. $\ Ax = b$

$$x \geq 0,$$

where the dimension of $A$ is $m \times n$, and its rows are linearly independent; $x, c, z$ are vectors of size $n$. We remark that an LP with inequality constraints can be transformed into the standard form by introducing extra slack variables (see Bertsimas and Tsitsiklis (1997)).

**Basic solution.** We first provide the formal definition of a basic solution to the LP. We say that a solution $x \in \mathbb{R}^n$ is a basic solution to the LP if and only if we have $Ax = b$ and there exist indices $B(1), \ldots, B(m)$ such that:

1. The columns of matrix $A$ indexed by $B(1), \ldots, B(m)$, which we denote as $A_{B(1)}, \ldots, A_{B(m)}$, are linearly independent;

2. If $i \notin \{B(1), \ldots, B(m)\}$, then $x_i = 0$.

In addition, we call $x_{B(1)}, \ldots, x_{B(m)}$ the basic variables and $x_i$ for $i \notin \{B(1), \ldots, B(m)\}$ the non-basic variables. We say that a basic solution $x \in \mathbb{R}^n$ is a basic feasible solution if it is also feasible (i.e., $x \geq 0$). We say that a basic solution $x \in \mathbb{R}^n$ is an optimal basic solution if it is both feasible and optimal to the LP.

**Simplex/dual simplex method.** It is known that if an LP in standard form has an optimal solution, then there exists an optimal basic solution. The simplex and dual simplex method are both algorithms designed to find the optimal basic solution $x^*$ to the LP. In the implementation of the simplex/dual simplex method, the algorithm keeps track of two sets of conditions:

1. **Optimality condition.** Let $x$ be a basic solution and let $B = [A_{B(1)}, \ldots, A_{B(m)}]$ be its associated basis matrix. Let $c_B = [c_{B(1)}, \ldots, c_{B(m)}]^\top$ be the vector of costs of the basic variables. For each $j \in [n]$, we define the reduced cost $\overline{C}_j$ of the variable $x_j$ as:

$$\overline{C}_j = c_j - c_B^\top B^{-1} A_j. \tag{39}$$

At a high level, for a non-basic variable $x_j$, the reduced cost of $\overline{C}_j$ measures the rate of cost change if we bring $x_j$ into the basis and move along the direction of $x_j$. Intuitively, if there exists an non-basic variable with positive reduced-cost, we should put it into our basis. Our optimality condition is thus defined as:

$$\overline{C} \leq 0.$$

Intuitively, if the optimality condition is satisfied at some solution $x$, then moving along any direction would only decrease our objective function.

2. **Feasibility condition.** Let $x$ and $B$ be its corresponding basis matrix; let $x_B = [x_{B(1)}, \ldots, x_{B(m)}]$. Since the columns of $B$ are independent and $Ax = Bx_B = b$, the feasibility condition is defined as:

$$x_B = B^{-1} b \geq 0.$$

This ensures that $x \geq 0$. 
In Theorem 3.1 of Bertsimas and Tsitsiklis (1997), it is shown that a basic solution that satisfies both the optimality and feasibility conditions is an optimal solution to the LP. Reversely, we also have that a nondegenerate, optimal basic solution to the LP must satisfy both the optimality and feasibility conditions.

Both the simplex method and the dual simplex methods are designed based on the optimality and feasibility conditions above. In the simplex method, the algorithm keeps a basis $B$ at each iteration, which corresponds to $m$ basic variables. At every iteration, the simplex method makes sure that the feasibility condition is always satisfied, and it swaps a basic variable with a non-basic variable that can increase the objective function. In the dual simplex method, the algorithm also keeps a basis $B$ at each iteration, which corresponds to a basic solution that might not be feasible. At every iteration, the dual simplex method makes sure that the optimality condition is always satisfied for its basic solution, and it swaps a basic variable with a non-basic variable such that the basic solution gets closer to the feasibility region. The simplex/dual simplex method terminates when both optimality and feasibility conditions are satisfied, which implies that given the current choice of basis $B$, an optimal basic solution has been found. See Bertsimas and Tsitsiklis (1997) for an extended discussion of simplex/dual-simplex method.