This work has been partially supported by MIAI @ Grenoble Alpes. (ANR-19-P3IA-0003)

Master Operations Research and Combinatorial Optimization
Master Informatique / Master Mathématiques & Applications

Online Algorithm With Prediction Information For The Bounded Allocation Problem

Eniko Donatella Toth

27 June 2022

Research project performed at LIG

Under the supervision of:

Kim Thang Nguyen, University Paris-Saclay, IBISC
Denis Trystram, University Grenoble Alpes, MIAI

Defended before a jury composed of:

Prof Nadia Brauner
Prof Van-Dat Cung
Researcher Nicolas Gast
Prof Kim Thang Nguyen
Prof Denis Trystram

June 2022

Abstract

Numerous research projects focus on online algorithms with predictions for different applications such as scheduling, caching (paging), clustering, and ski rental. Bamas, Maggiori, and Svensson [1] recently provided a primal-dual framework for linear covering-packing problems aiming for a unified approach. They extended the online primal-dual method by incorporating predictions to achieve performance beyond the worst-case case analysis. We follow this line of research and propose an algorithm to improve on known performance bounds for the bounded allocation problem and show supporting experiments.

Résumé

De nombreux projets de recherche se concentrent sur des algorithmes en ligne pour différentes applications comme l'ordonnancement, la gestion de caches, le clustering et la location de ski. Bamas, Maggiori et Svensson [1] ont récemment proposé une approche primale-duale pour les problèmes de recouvrement-rangement avec l'objectif de construire une approche unifiée. Ils ont fait une extension de la méthode primale-duale en-ligne en y incorporant des prédictions pour atteindre des résultats supérieurs au pire cas donné par l'analyse. Nous poursuivons cette voie de recherche et proposons un algorithme pour améliorer le bourne de performance connue pour le problème de allocation bornée et montrons quelques résultats expérimentaux encourangeants.

Contents

1 Introduction

1.1 Background

Digitalization and complexity. The industrial world has been revolutionized by digitalization, influencing many aspects of our daily lives [12]. Computing technology has shrunk in size and increased in efficiency throughout the previous century, while production costs have lowered to match the average person's budget. As a result, the number of digital devices has skyrocketed (page 21 of [12]). The Internet of Things (IoT) era has come, and we now live in a linked world with sensors and smart fridges. Humanity relies on digital technologies to interact with the Internet. This connection generates a previously unfathomable amount of data, posing numerous problems to the scientific community in dealing with the underlying system's complexity [9].

The digital transformation impacted the industry at least as much as our day to day lives. The use of computers and digitalization paved the way for new industrial processes. Industry 4.0, or the fourth industrial revolution, among numerous things, focuses on advanced automation techniques, such as self-automating systems with machine learning [6]. Digital equipment provides consequential metrics regarding the equipment's state. These metrics enable fine-grained control and contribute to a better understanding of the equipment's system through data analytics. Unfortunately, this increase in data and the number of devices to handle can quickly become uncontrollable and extremely complex. As a result, machine learning techniques enjoyed increasing popularity in the last twenty years.

However, machine learning is inapplicable in some cases, for example, when mathematical performance guarantees are indispensable. Therefore, it is crucial to study the impact of the ever-growing digitalization on different optimization techniques and algorithm design. Digitalization increases the system's complexity in which the algorithms execute. For example, due to the ever-increasing amount of data with which the algorithms must cope. In addition, as the number of devices grows, so does the level of complexity. As an example, we can consider the increased complexity imposed by the type and number of connections between devices in a distributed algorithm as the number of devices grows.

Human interaction and uncertainty. Apart from the increased system complexity, algorithm designers face a second challenge: uncertainty. The model of a problem may contain probabilistic features due to many reasons [13]. Firstly, most use cases involve human-machine interactions and are naturally uncertain as an effect of human behaviour. Secondly, even if, in theory, we could model a specific problem deterministically, it may impose tractability issues due to the high amount of required computation. Additionally, it might impose difficulties on reasoning about the model's properties. Lastly, some complex models contain uncertainty by nature. We can consider examples such as the uncertainty of measurements due to machine imprecisions, the uncertainty of the order of completion of concurrent events, the uncertainty of the delay time to receive a TCP packet over the network, or any naturally occurring unexpected events in a system.

Explorable uncertainty. There are many ways to model uncertainty. In this work, we consider online algorithms, where the order and type of arriving events are uncertain. Even though online algorithms capture future uncertainty realistically, traditional online algorithms do not exploit available information to its full potential. Let us consider the example of an exam at a university. Even though we do not know the students' grades before the exam, we can have meaningful assumptions. We can estimate the grades by observing the students' performance during the semester or build a statistical model from the previous years' exam results. Leveraging such information is crucial to improve known best performance guarantees of online algorithms. We can explore the uncertainty of these kind of problems with several techniques, such as predictions of oracles that learn from previous data, forecasts that predict the distribution of the events, or with different stochastic models.

To summarise the background of this project, the environment we consider for our algorithms is complex and uncertain. We aim to create realistic and tractable models, design algorithms that can handle the previously mentioned challenges, and improve on known competitive bounds by incorporating explorable information in uncertain settings.

1.2 Motivation

Buchbinder and Naor [3] showed several recent algorithms that exploit the primal-dual method, an elegant framework for designing online algorithms. The primal-dual technique represents the dynamism of the online problem by casting it to a linear program formulation and revealing the constraints gradually upon each arriving event. After each decision, the algorithms need to maintain feasible solutions to both the primal and the dual formulations. Afterwards, we can use the weak duality theorem to obtain the competitive ratio of the algorithm by bounding the change of the primal objective value with the dual objective value.

Data-driven machine learning methods improved significantly in the last few decades. Among the numerous applications, Bamas, Maggiori, and Svensson [1] proposed a new algorithm design technique focusing on meta-algorithms that combine state-of-the-art online algorithms with machine learning predictions. The challenge is to find the right balance between the components and provide performance guarantees even without any assumption on the predictions. Ideally, we would like to improve the algorithm's performance when the prediction is correct while keeping the same worst-case guarantees as the algorithm had before the predictions when it is incorrect.

The bounded allocation problem (or bounded online allocation) is an extensively studied problem, which already has a traditional online algorithm (see Chapter 13 of [3]) that performs close to the best possible competitive ratio $(1 - {}^{1}/_{e})$. To go beyond its known performance bound, we need to obtain and integrate additional information to the algorithm's decision making. In this report we propose a meta-algorithm that integrates the predictions of a black-box oracle with the decisions of the traditional online algorithm. With sufficiently accurate predictions, our algorithm can surpass the performance of the original algorithm, that we present in the experiment section of this report. Online allocation is an abstract problem that has numerous practical applications, such as advertising, resource allocation, or bandwidth management on a network. Additionally, these are high-impact applications where slight performance improvements can result in significant gains in the given use case.

1.3 Related work

Primal-Dual method. Linear programming has demonstrated to be an essential optimization method in many industrial areas. Beyond finding optimal solutions for offline combinatorial problems, it has proven to be a valuable tool for online algorithm design. Buchbinder and Naor [3] created a comprehensive study of the primal-dual method and demonstrated its applicability through numerous well-known problems. The primal-dual method is a powerful technique both for designing and analyzing online algorithms. It has found to be highly effective in various combinatorial problems, especially for approximation algorithms of NP-hard problems.

The primal-dual method relies on the linear program formulation of the combinatorial problems. We can express the primal with a general model as follows.

$$\begin{aligned} \text{(Primal)}\ &: & \min \sum_{i=1}^{n} c_i\ x_i \\ \text{s.t.}\ \ \forall\ 1 \le j \le m\ &: & \sum_{i=1}^{n} a_{ij}\ x_i \ge b_j \\ \text{and}\ \ \forall\ 1 \le i \le n\ &: & x_i \ge 0 \end{aligned}$$

The variable of the problem is the vector x, while the vector c corresponds to some cost related to the variables. The matrix a and the vector b expresses the constraints of the problem. The pair of the primal problem is the dual. It has m dual variables that correspond to the primal constraints and n constraints that correspond to the primal variables. We can formulate the dual as follows.

$$\begin{aligned} \text{(Dual)}\ &: & \max \sum_{j=1}^{m} b_j\ y_j \\ \text{s.t.}\ \ \forall\ 1 \le i \le n\ &: & \sum_{j=1}^{m} a_{ij}\ y_j \le c_i \\ \text{and}\ \ \forall\ 1 \le j \le m\ &: & y_j \ge 0 \end{aligned}$$

A well-known theorem in linear programming is the *theorem of weak duality.* We recall that the theorem states that any feasible solution of the dual bounds any feasible solution of the primal. Since in our formalization the primal problem is a minimization problem, the dual provides a lower bound. Formally,

$$\sum_{i=1}^{n} c_i\ x_i \ge \sum_{j=1}^{m} b_j\ y_j$$

We can construct a constant time upper bound from the dual, using the classical *approximate complementary slackness theorem.* Let us consider a feasible solution for both the primal and the dual (x and y), such that they satisfy the following conditions.

- For $\alpha \ge 1$ and $\forall\ 1 \le i \le n$ if $x_i > 0$, then $\frac{c_i}{\alpha} \le \sum_{j=1}^{m} a_{ij}\ y_j \le c_i$.
- For $\beta \ge 1$ and $\forall\ 1 \le j \le m$ if $y_j > 0$, then $b_j \le \sum_{i=1}^{n} a_{ij}\ x_i \le b_j\ \beta$.

Then, according to the *approximate complementary slackness theorem*, the following holds and we obtain a constant time upper bound.

$$\sum_{i=1}^{n} c_i \; x_i \leq \alpha \; \beta \; \sum_{j=1}^{m} b_j \; y_j$$

So far, we have described the linear program formulation of an offline setting, where every detail of the problem is known in advance. We can represent an online problem with the linear program formulation by gradually revealing the variables and corresponding constraints. Then, one can rely on the primal-dual approach and design an online algorithm that builds feasible solutions for both the primal and the dual problems throughout the execution. Casting a problem to a linear program formulation may help to identify the difficulty of the online setting. Additionally, the dual problem provides a natural way to analyze the performance of the online algorithm.

Learning augmented algorithms. There exist many efficient algorithms to solve known combinatorial problems. These algorithms often obtain the best theoretically possible competitive ratios or perform close to these bounds. In order to further improve their performance, we need additional information. Bamas, Maggiori, and Svensson [1] proposed a framework based on the primal-dual method to create meta-algorithms that combine known efficient algorithms with prediction information. The principle of these meta-algorithms is to outperform the original algorithms when the predictions are correct but maintain close worst case guarantees when the predictions are incorrect.

We focus on online algorithms, where traditional algorithms tend to be overly cautious due to future uncertainty. As a result, the algorithms' performance is often far from what meta-algorithms can achieve with predictions. Furthermore, many real-life use cases have predictable parts of uncertainty, and the online events follow patterns we can exploit with machine learning techniques. The challenges of these meta-algorithms are the following.

- identifying what is predictable in the problem
- finding how to incorporate the prediction information with the traditional algorithm
- obtaining both good consistency with the prediction and good worst case guarantees
- achieving slow performance degradation with increasing prediction error rates

Algorithms using the learning augmented primal-dual framework build feasible solutions for both the primal and the dual problems while incorporating in the decision making the prediction with an adjustable weight based on the amount of trust the algorithm has in the prediction. The algorithm we propose in this report relies on this framework, therefore Section 3 provides a concrete example and further explanations.

Water-filling algorithm. Narrowing further our context, let us look at the problem of this report. The bounded allocation problem is a packing problem that generalizes the bipartite maximum matching problem. In the online bipartite maximum matching problem, we have a graph $G = ((L, R),\ E)$ whose vertex set is partitioned into two sets L and R. We assume that the algorithm knows the vertices in L from the beginning of the execution. The vertices in R are revealed gradually step by step in an online fashion. The objective is to find a maximum matching in this graph. Without any further assumptions, the best known online algorithm to solve the fractional online bipartite maximum matching problem is the water-filling algorithm. Among others, Kleinberg [7] provides a clear definition of the water-filling algorithm and proves the algorithm's competitive ratio.

During its execution, the water-filling algorithm assigns a weight $w_e \in [0,1]$ for each edge $e \in E$ of the graph. The weights correspond to the contribution fraction of the edges to the matching. Since we are considering a fractional matching, for each vertex v the sum of the weights on the edges connected to v can not exceed 1. Formally,

$$\forall\ v \in \{L \cup R\}\ :\ \sum_{e \in \delta(v)} w_e \leq 1$$

where $\delta(v)$ corresponds to the edges connected to v. The general principle of the water-filling algorithm is to balance assignments on the edges keeping the state of the instance into account. For each online vertex $v \in R$, the algorithm assigns fractionally and continuously weights on the edges of $\delta(v)$ where the sum of the weights of the offline vertices is minimal. Formally, for each $e = (v_i, v) \in \delta(v)$ where $v_i \in L$, the algorithm assigns a small fraction ϵ on the edges where $\sum_{e \in \delta(v_i)} w_e$ are minimal. By using the primal-dual method we can obtain that the water-filling algorithm has a competitive ratio of $1 - {}^1\!/_e$.

Ad-auction. The ad-auction problem is the closest problem in the literature to the bounded allocation problem. It is a special case of the online maximum matching problem. The input of the algorithm is n buyers with their corresponding budgets B_i. We can consider the buyers as the left hand side of the bipartite graph, the vertices in L. In the ad-auction problem m items arrive online. For each arriving item j, each buyer can propose a bid b_{ij} to buy the item. Items are analogous to the right hand side of the bipartite graph. The algorithm decides which buyer to sell the item to. The objective is to maximise the revenue of the seller. Buchbinder, Jain and Naor [4] provides an efficient online algorithm to solve the ad-auction problem. Their initial algorithm builds on the primal-dual method and assigns the item to the buyer that maximizes the proposed bid times the available budget of the buyer. They obtained a competitive ratio of

$$(1 - {}^1\!/_c)(1 - R_{\max}) \ \text{ where } \ c = (1 + R_{\max})^{({}^1\!/_{R_{\max}})}$$

and $R_{\max}$ stands for the maximum ratio between any bid and the corresponding buyers budget. When this ratio tends to 0, the competitive ratio of the algorithm tends to $1 - {}^1\!/_e$. Therefore, we obtain a similar competitive ratio as for the online maximum matching problem.

Buchbinder, Jain and Naor [4] proposed an other algorithm which incorporated stochastic information to their initial online primal-dual algorithm. The additional information that the algorithm gains are the lower bounds on the fraction of spent budget that each buyer is expected to spend. Since the algorithm uses the primal-dual method, it builds feasible solutions to both the primal and dual problems. The intuition behind the algorithm is to augment the way the algorithm assigns values to the dual variables, gaining a tighter bound on the increase of the objective value. In our report we rely on a similar intuition in our proposed algorithm. We discuss the primal and dual variables in later sections. The improved competitive ratio is

$$\left(1 - \frac{1-g}{c^{(1-g)}}\right)(1 - R_{\max})$$

where c is the same constant that we defined above and g is the minimum bound on the expected spent budget fractions.

Nguyen and Durr [11] presented a different learning augmented primal-dual algorithm to solve the ad-auction problem. Their meta-algorithm combines the predictions of a black-box oracle with the initial primal-dual algorithm of [4]. The principle of the meta-algorithm is to split the assignment between the two components by choosing a parameter $\eta \in [0,1]$ and allowing the primal-dual algorithm to assign η fraction, while the prediction to assign $(1-\eta)$ fraction. The parameter η corresponds to the amount of doubt the algorithm has in the prediction. If $\eta = 0$, there is no doubt and if $\eta = 1$, there is complete doubt and the meta-algorithm only uses the integrated primal-dual algorithm. In our report we also rely on a black-box oracle and use the same η parameter. The obtained competitive ratio of the meta-algorithm is

$$\left(1 - \frac{1}{C}\right) \frac{1}{1 + R_{\max}} \quad \text{and} \quad C = (1 + R_{\max})^{(\eta/R_{\max})}.$$

Controlling uncertainty. The ad-auction problem was an excellent example of how different stochastic methods and predictions can be used to augment state-of-the-art deterministic algorithms. Modelling uncertainty and future events is an ongoing challenge in the research community. It is crucial to find the right balance between keeping the models tractable and realistic.

Kumar, Purohit, Schild, Svitkina and Vee [8] modelled the uncertainty in online biparite maximum matching by treating one part of the online vertices as predictable and the other as adversarial. The algorithm does not know the ratio of the predictable and adversarial vertices. However, since the algorithm knows all the predictable online vertices and the offline vertices ahead of execution, we consider this scenario as a semi-online setting. The authors of [8] propose the `RANKING` algorithm to solve the semi-online biparite maximum matching problem. The `RANKING` algorithm consists of a preprocessing and an online phase. In the preprocessing phase, the algorithm finds a random maximum matching between the offline and the predictable online vertices. Additionally, it creates a random permutation of the offline vertices which do not participate in the randomly chosen maximum matching. We call these the reserved vertices. In the online phase, for each arriving vertex v, if v is a predictable vertex, the `RANKING` algorithm matches it according to the maximum matching chosen in the preprocessing phase. If the new vertex v is not predicted, it is matched to the first vertex among the reserved vertices that v is neighboured with. The `RANKING` algorithm has a competitive ratio of

$$(1-\delta) + ({\delta^2}/{2})(1 - {1}/{e})$$

where $\delta = 1 - {f(H)}/{f(G)}$, where $f(H)$ is the size of the matching obtained during the preprocessing phase and $f(G)$ the size of the final matching at the end of the execution. The closer δ is to 0, the closer the solution to the optimal offline solution due to the predicted information.

Esfandiari, Korula and Mirrokni [5] present algorithms that combine stochastic and online algorithms that can adaptively react to inaccurate predictions. In many real-life use cases, the online events can be predicted with good precision due to a large amount of available historical data. However, if the predictions are incorrect, stochastic algorithms have a slow reaction time to adjust to the discrepancy. The authors proposed another uncertainty model for the online ad-auction problem. Similarly to the semi-online model, they consider an adversary as well. However, they rely on a forecast instead of a prediction, containing a distribution of item types (for example, an item with n proposed bids) and the number of expected items. During the execution, their algorithm checks if the new item aligns with the expected distribution. If yes, the algorithm allocates the item optimally according to the expected distribution. If not, the algorithm assumes that the item is adversarial and only allocates it if some remaining fraction is available outside the forecasted item allocations. Ideally, their algorithm would first allocate the forecasted items and then the adversarial ones if there are still possible assignments. However, the items arrive in random order. Therefore, the algorithm reserves allocations for the forecasted items and recognizes these items by verifying that the observed distribution with the arrival of the new item is at most ϵ-far from the expected distribution.

Other analysis methods. Every previously discussed method revolves around finding the proposed algorithms' competitive ratio with worst-case analysis. Worst-case guarantees establish the general (input independent) performance of algorithms and helps to provide accurate advice on whether a given algorithm suits a given use case. Additionally, it is often easier to consider a worst-case scenario than to determine the average performance over the possible inputs. However, the worst-case analysis gives an inaccurate performance prediction when the algorithms have a significant performance variation across inputs of a given size. As a result, for many complex problems, worst-case guarantees are inaccurate and overly pessimistic. To address this issue, researchers have been aiming to establish alternative methods for algorithm analysis. Roughgarden organized these efforts in a book titled *Beyond the worst case analysis* [2]. As an example, Section 1.3 of the book describes one possible alternative: choosing appropriately fine-grained parameterization of the input space. This method allows us to compare algorithms with parameterized guarantees. We can consider these parameters as additional information on the input other than its size. We can combine several analysis methods as well. Worst-case analysis can highlight the problem's most difficult instances, while parameterized analysis can provide detailed and rich information about the algorithm's performance on specific problem inputs.

1.4 Content and contribution

Section 2 of the report dives into more details about the bounded allocation problem and contains a dummy example. Section 3 describes the intuition behind our proposed algorithm, explains how the algorithm works, and shows the execution through another dummy example. Section 4 presents the performance guarantee proofs. Our main result is Theorem 4.4, which states that our proposed meta-algorithm for the bounded allocation problem is $(1-\eta)$-consistent with the prediction and

$$1/\left(\frac{e}{e-1}+\frac{(1-\eta)(1-e^{\eta-1})\ e}{e-1}\right)\text{-competitive}$$

with the optimal offline solution, where $\eta \in [0,1]$ is the parameter of the algorithm controlling the doubt in the prediction. Section 5 walks us through the implementation of the algorithm and the experimental results. Finally, in Section 6 we conclude and mention possible future work on the subject.

2 The Problem

2.1 Context

We model the bounded allocation problem as a generalization of the online bipartite maximum matching problem. In this problem, we consider two categories which we can represent with the vertices of a bipartite graph. The items in one of the categories arrive online. The algorithm aims to find a maximum matching in the gradually revealed graph. In the bounded allocation problem, we do not require a one to one matching between vertices of the two categories.

As a real-life example, we can consider ad allocation, where the algorithm attempts to sell as many ad slots as possible. In this case, the categories of the bipartite graph are buyers and ad spaces. Each buyer has a budget, and the algorithm knows about it from the beginning of its execution. When an ad slot becomes available, the buyers discover its price and decide if they are interested in purchasing. Afterwards, the algorithm makes an irrevocable decision regarding whom to sell the ad space. The bounded allocation problem is a special case of the ad-auction problem, in which buyers propose their bid to buy the item instead of a fixed price. Additionally, we can consider the online vertices to be splittable. A real-life fractional example of the bounded allocation problem is bandwidth allocation in a communication network.

2.2 Example

Let us walk through a concrete example assuming the items are not splittable. We can consider a simple greedy algorithm that assigns each item to the buyer with the smallest ID such that it has enough available budget to purchase it. Figure 1 shows the initial state of the problem at the beginning of the execution. We have four buyers, each with a budget visible above their icon. The algorithm assigns the first item to the third buyer following the greedy logic. The algorithm can not sell the second item as no interested buyer has enough available budget. Finally, the fourth buyer gets the third item, and the execution stops. Figures 2-4 show the execution steps after each item. To avoid clutter, we only keep edges between items and buyers from previous steps if the buyer purchased the item. Clearly, our simple greedy algorithm is not optimal. However, it highlights the problem's difficulty: regret of previous decisions.

2.3 Formal definition

The bounded allocation problem consists of n buyers and m items. Each buyer $1 \leq i \leq n$ has a fixed independent budget B_i that the algorithm knows. The items arrive one by one in an online fashion. Each item $1 \leq j \leq m$ has a fixed independent price b_j and a subset of buyers S_j interested in purchasing. We consider the items to be splittable. Upon each arriving item, the algorithm makes an irrevocable decision regarding whom to sell and in which fraction. The objective is to maximize the profit of the seller. Table 1 lists the notations of the problem.

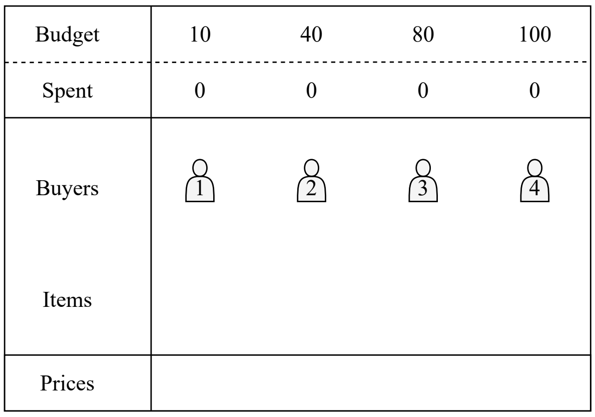

Figure 1: Initial state

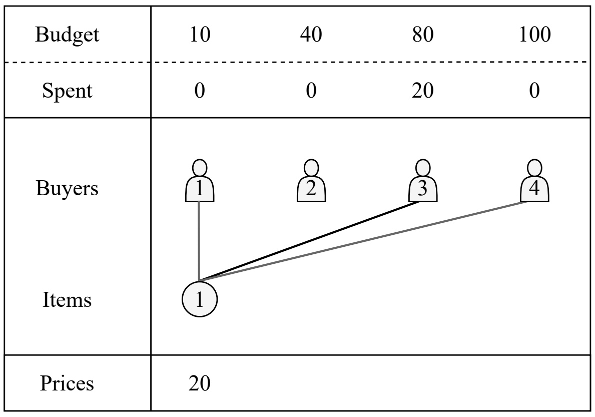

Figure 2: State after the first item

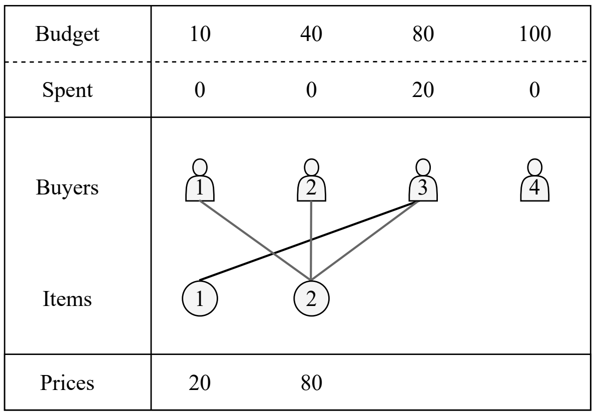

Figure 3: State after the second item

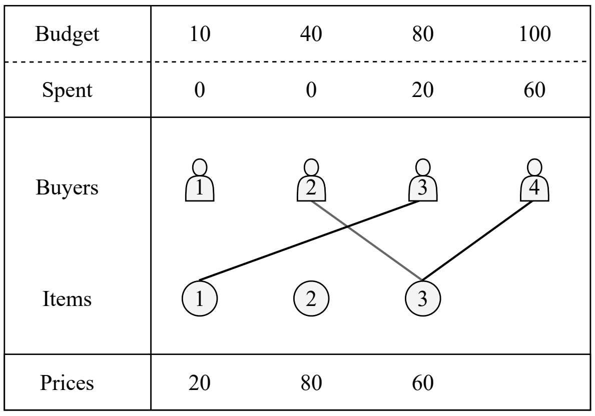

Figure 4: State after the third item

Figure 5: Execution steps of the dummy example

Symbol	Description
n	number of buyers
m	number of items
i	index of a buyer
j	index of an item
B_i	budget of buyer i
b_j	price of item j
S_j	interested buyers of item j
d	maximum size of S_j
x_{ij}	fraction of item j assigned to buyer i

Table 1: Problem notations

3 The New Meta-Algorithm

3.1 Water-filling

Looking back on the dummy example in Figures 1-4, we can observe that if the algorithm assigned item one to the fourth buyer, it would have obtained an optimal solution. The natural intuition is to assign the items to buyers with the most available budget, potentially reserving some budget of each buyer for future purchases. In the fractional variant of the problem, this method is particularly fruitful as we can split items with a high price among several buyers. The best known fractional online algorithm for the bounded allocation problem is the water-filling algorithm, which exploits this intuition and continuously assigns fractions of the arrived item to the buyers who spent the slightest fraction of their budget. This report builds on a variation of the water-filling algorithm (described in Section 13 of the survey of Buchbinder et al. [3]), assuming that the number of interested buyers for the items is bounded. The algorithm uses this bound to create level sets to group buyers based on their spent budget fraction and splits the arriving items equally among interested buyers in the lowest level set.

3.2 Description

We propose a meta-algorithm that combines the mentioned water-filling algorithm and a black-box oracle that predicts whom to sell the current item. In our setting, the predictions are integral, but the solution of the meta-algorithm is fractional. For example, for item j, the prediction is $pred(j) = i^*$, a single predicted buyer i^*. Since some instances do not have an optimal integral solution, the solution produced by the oracle is not necessarily optimal. We chose to rely on integral predictions to have a simple model and to ease the difficulty of the performance proofs. The parameter $\eta \in [0, 1]$ influences the ratio with which the meta-algorithm considers the prediction, where zero means no doubt in the prediction, while one stands for complete doubt. Algorithm 1 below shows the high-level description of our meta-algorithm.

Algorithm 1: Algorithm with Prediction for the Bounded Allocation Problem.

```
1  Initialize the variables and the level sets
2  for each new item $j$ do
3      Let $i^*$ be the predicted buyer of item $j$, formally, $pred(j) = i^*$
4      while there exists $i \in S_j$ who spends less than $\eta$ fraction of its budget do
5          Continuously allocate fractions of $j$ to buyers on the lowest levels        // Stage 1
6
7      repeat
8          Continuously allocate fractions of $j$ to $i^*$                              // Stage 2
9      until either:
             • $(1 - \eta)$ fraction has been allocated to $i^*$
             • or $j$ is completely sold
             • or $i^*$ exhausted its budget
10
11     while there exists $i \in S_j$ with non-exhausted budget and $j$ is not completely sold do
12         Continuously allocate fractions of $j$ to buyers on the lowest levels        // Stage 3
```

After initializing the level sets, the algorithm assigns each arriving item. The algorithm consists of three stages. The first contains the limit assignments, to guarantee that each interested buyer spends at least η fraction of its budget before allowing prediction assignments to occur. The second consists of the prediction assignment. By construction of the algorithm, once all the interested buyers reached the η limit in Stage 1, the algorithm assigns up to $(1 - \eta)$ fraction of the item to the predicted buyer, keeping the remaining fraction and the available budget into account (line 9 of Algorithm 1). The third stage is the equal assignment of the remaining fraction (if there is some) among the interested buyers on the lowest levels. The assignments of Stages 1 and 3 correspond to the water-filling technique of the original algorithm of [3].

If $\eta = 1$, the algorithm ignores the prediction information. If $\eta = 0$, the items are assigned according to the prediction until a feasibility issue occurs. Suppose the prediction is partially incorrect and tries to allocate more items to a buyer than its budget allows. In such a case, the algorithm attempts to allocate the excess fraction to the other interested buyers. Therefore, it mitigates the errors of the prediction to some extent.

3.3 Component observations

In the following, we examine the components' behaviour inside the meta-algorithm and show possible risks in their decision making. Additionally, we provide some insights into the logic of the meta-algorithm.

There are two pitfalls in decision making. Firstly, the algorithm might not assign enough fractions of items to a buyer who is only interested in a few of them. We can theoretically find such suboptimal decisions by analyzing the solution of the algorithm at the end of the execution. (Even though for large instances, we can not find such improvements efficiently.) As a result of these suboptimal decisions, we could improve the solution of the algorithm, more precisely, sell more fractions of some items to the buyers by rearranging the assignments. It is possible to make such an improvement if the following conditions hold.

- there exists a buyer i^* with a non-exhausted budget at the end of the execution, formally,
  $\sum_j \; b_j \; x_{i^*j} < B_{i^*}$
- there exists an item that buyer i^* purchased in some fraction, but not completely,
  $\exists \; i'$ s.t. $i^* \in S_{j'}$ and $0 < x_{i^*j'} < 1$
- there exists a subset of buyers with exhausted budgets at the end of the execution, formally,
  $\exists \; R \neq \emptyset$ s.t. $\forall \; i \in R : \; \sum_j \; b_j \; x_{ij} = B_i$
- there exists a reassignment of items between the buyers, such that the objective value is unchanged and afterwards we have a buyer with exhausted budget who purchased some fraction of item j' and an other item which was not sold completely, formally,
  $\exists$ a reassignment of items such that
  $\exists \; \hat{i} \in R$ and $\hat{i} \in S_{j'} \setminus \{i^*\}$ s.t. $x_{\hat{i}j'} > 0$ and $\exists \; \bar{j}$ s.t. $\hat{i} \in S_{\bar{j}}$ and $\sum_i \; x_{i\bar{j}} < 1$

If the above conditions hold, we can improve the solution of the algorithm by decreasing the amount buyer $\hat{i}$ received from item j' and instead assign more fraction of item $\bar{j}$ to this buyer. We can use the freed fraction of item j' to assign it to buyer i^*. Formally, we find $0 < \alpha \leq x_{\hat{i}j'}$ and $0 < \beta \leq 1 - \sum_i x_{i\bar{j}}$ such that $\alpha\ b_{j'} = \beta\ b_{\bar{j}}$ and $\sum_j\ b_j\ x_{i^*j} + \alpha\ b_{j'} \leq B_{i^*}$. The second pitfall complements the first one, corresponding to the case when the algorithm assigns too much fraction of items to a buyer that is interested in many items.

The meta-algorithm consists of two components which represent opposing strategies. On one side, the water-filling algorithm creates a fractional solution and mitigates the risk of not being able to sell an item that occurs when all interested buyers exhaust their budget, but it is prone to make suboptimal decisions described before as the first pitfall. On the other side, the prediction suggests an integral solution that imposes a higher risk of not being able to sell an item and more likely to make decisions corresponding to the second pitfall. Nevertheless, in real-life scenarios, we consider the prediction to have high accuracy, and therefore it can be beneficial for the algorithm to rely on it. To find the right balance between the components, the meta-algorithm first guarantees that all buyers spend η fraction of their budget to lower the risk of the prediction assignments. We detail the benefit of Stage 1 assignments in Section 4 in the robustness part of the performance analysis.

3.4 Example

Let us observe the execution of the meta-algorithm through an example. We follow a similar setup as in the previous dummy example but allow fractional solutions and enable the prediction. On the top segment of the tables on Figures 6-9, we show each buyer's index, available and already spent budget. The bottom of every table shows each item's index, price, and corresponding prediction. In the middle segment, each vertical bar corresponds to the buyer's budget in that column. We fill the bars in proportion to the buyer's spent budget. We highlight each item's assignment with colours in the buyer's budget bar.

First item. The algorithm assigned the first item entirely in Stage 1 using the water-filling algorithm to satisfy the η fraction of spent budget condition. After the assignment of the first item, buyer 1, 3 and 4 spent 0.2 fraction of their budget, therefore they reached the η limit. The state of the instance after the first item is visible on Figure 6.

Second item. The predicted buyer for the second item was buyer 3. Even though this buyer reached the η limit in the previous step, buyer 2 was below the limit. Therefore, the algorithm first assigned $^4/_{30}$ fraction of item 2 to buyer 2 in Stage 1. Afterwards, the algorithm assigned $(1-\eta)$ fraction, precisely $^{24}/_{30}$, to buyer 3 in Stage 2. Finally, the remaining $^2/_{30}$ fraction was assigned in Stage 3 using again the water-filling algorithm. Buyers 1 and 2 received some small fractions of item 2 in Stage 3. The state of the instance after the second item is visible on Figure 7.

Third item. At the beginning of the third item's assignment, all buyers spent η fraction of their budget, so Stage 1 was skipped. Afterwards, the algorithm assigned $(1-\eta)$ fraction of the third item to buyer 4. Since η is small in this example, the remaining fraction was assigned to the second buyer. The state of the instance after the third item is visible on Figure 8.

Fourth item. In this specific example, due to water-filling in the previous steps, the algorithm is not able to sell item 4 completely. The optimal objective value is 90, while the objective value of the algorithm is 79.86, achieving 88.73% of the optimum. The final state of the instance is visible on Figure 9 .

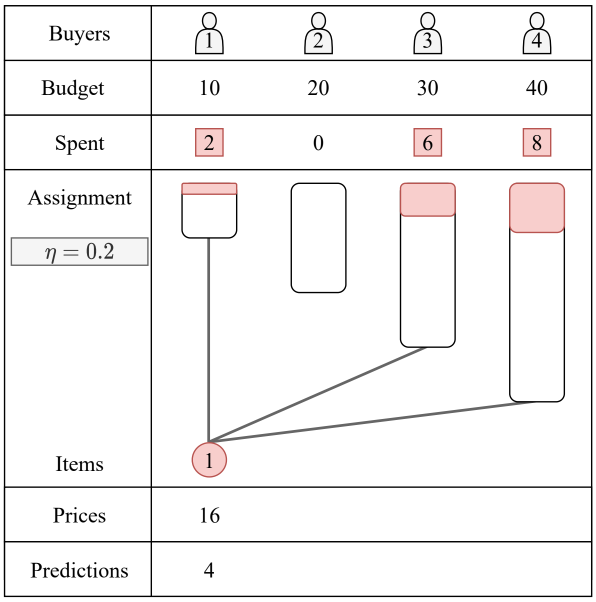

Figure 6: State after the first item

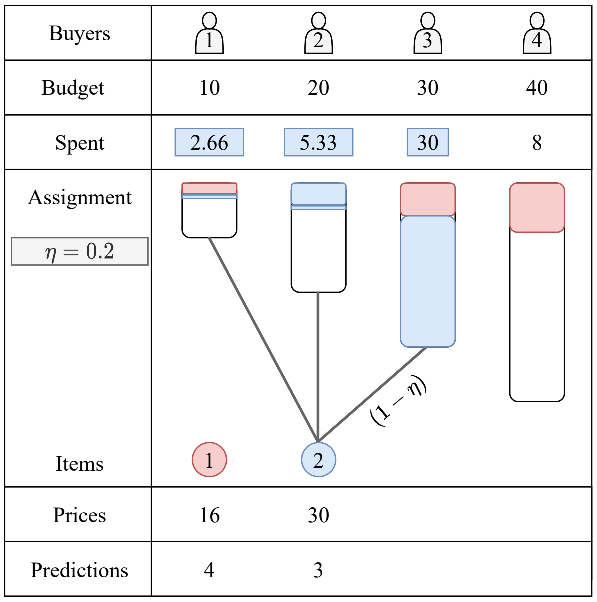

Figure 7: State after the second item

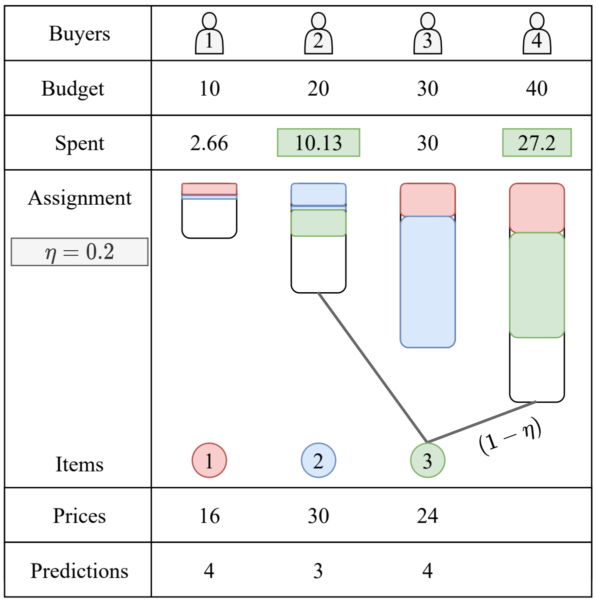

Figure 8: State after the third item

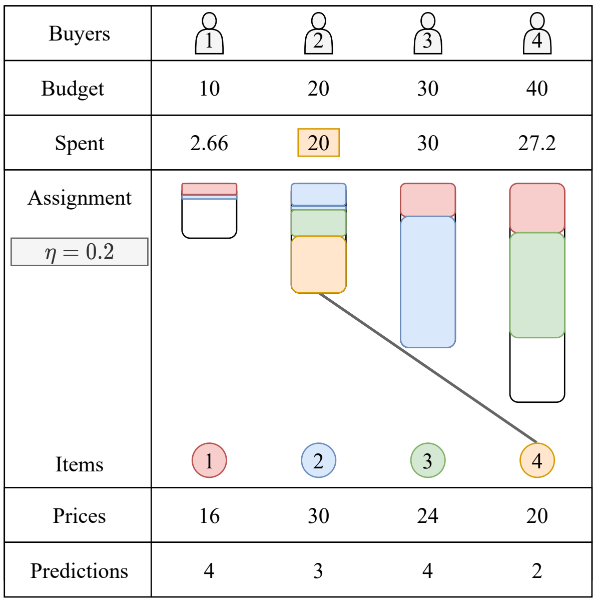

Figure 9: State after the fourth item

4 Performance Analysis

We use consistency and robustness (competitive ratio) metrics to determine the performance of our algorithm compared to the standard offline algorithm. An algorithm A is $c(\eta)$-consistent and $r(\eta)$-robust, if for all instances I of the problem, the following holds:

$$A(I) \geq \max\{c(\eta) \cdot P(I),\ r(\eta) \cdot O(I)\}$$

where $A(I)$, $P(I)$ and $O(I)$ are respectively the objective values obtained by the new algorithm, the prediction alone, and the optimal offline fractional solution on instance I.

4.1 Consistency

Lemma 4.1 (Consistency). *The new algorithm is $(1-\eta)$-consistent with the prediction.*

Proof. Let V be the set of buyers who exhaust their budget at some point during the execution of the algorithm. Let U be the set of items whose buyers did not exhaust their budgets, formally, $U = \{j : \texttt{pred}(j) \notin V\}$. We can formulate the total gain of a feasible prediction:

$$\sum_{\substack{j \\ \texttt{pred}(j) \neq \emptyset}} b_j = \sum_{\substack{j \\ \texttt{pred}(j) \in V}} b_j + \sum_{\substack{j \\ \texttt{pred}(j) \notin \{V \cup \emptyset\}}} b_j \leq \sum_{i \in V} B_i + \sum_{j \in U} b_j \tag{1}$$

where the inequality holds since any feasible allocation, including the prediction, can allocate items of value at most B_i (the complete budget) to buyer i.

Let $j \in U$ be an arbitrary item. By construction of the algorithm and the fact that the predicted buyer for item j does not exhaust its budget, the following can occur. The algorithm assigns item j entirely during the limit assignment (to satisfy the condition of η fraction spent by each interested buyer), or otherwise allocates the remaining fraction of item j to its predicted buyer i up to $(1-\eta)$ fraction. Note that the algorithm can always assign the remaining fraction since buyer i does not exhaust its budget by the end of the execution. In any case, we sell item j in at least $(1-\eta)$ fraction, therefore the gain of the algorithm on j is at least $(1-\eta)\ b_j$.

Observe that in the algorithm before the prediction assignment the total value of items in U sold to buyers in V is at most $\sum_{i \in V} \eta B_i$ due to the limit assignment rule. Note that this means that each predicted buyer i^* where $i^* \notin V$ gets at least

$$(1-\eta) - \left(\frac{\eta\ \sum_{i \in V} B_i}{b_j}\right)$$

fraction of item j. Therefore, the total gain of the algorithm is at least

$$\sum_{i \in V} B_i + \sum_{j \in U} (1-\eta)\ b_j - \eta \sum_{i \in V} B_i \tag{2}$$

where the first term is the gain of the algorithm restricted to buyers in V, the second one is the gain of the algorithm restricted to items in U and the last term is the upper bound of the total value of items in U sold to buyers in V before the prediction assignment. By (1) and (2), the gain of the algorithm is at least $(1-\eta)$ that of the prediction. $\square$

4.2 Robustness

To determine the robustness of the new algorithm, we relied on the primal-dual schema and the competitive ratio proof of the original algorithm of [3]. The linear program formulations align with the covering and packing problems. The bounded allocation problem corresponds to the packing problem. The linear programs are:

Primal (packing)

$$\begin{array}{rcrl} & \max & \sum_{j=1}^{m}\sum_{i\in S_j} b_j\ x_{ij} & \\ \forall\ 1\le i\le n & : & \sum_{j\mid i\in S_j} b_j\ x_{ij} & \le B_i \\ \forall\ 1\le j\le m & : & \sum_{i\in S_j} x_{ij} & \le 1 \\ \forall\ i\ \forall\ j & : & x_{ij} & \ge 0 \end{array}$$

Dual (Covering)

$$\begin{array}{rcrl} & \min & \sum_{i=1}^{n} B_i\ y_i + \sum_{j=1}^{m} z_j & \\ \forall\ j,\ i\in S_j & : & b_j\ y_i + z_j & \ge b_j \\ \forall\ i & : & y_i & \ge 0 \\ \forall\ j & : & z_j & \ge 0 \end{array}$$

Following the proof of the original algorithm in [3], we define a piece-wise linear function $f_d : [0,1] \to \mathbb{R}_{\ge 0}$ such that $f_d(1) = 1$ and for $0 \le u < 1$, if $\frac{\ell-1}{d} \le u < \frac{\ell}{d}$ for some $1 \le \ell \le d$ then

$$\begin{aligned} f_d(u) &= da_\ell\left(u - \frac{\ell-1}{d}\right) + da_{\ell-1}\frac{1}{d} + da_{\ell-2}\frac{1}{d} + \ldots + da_1\frac{1}{d} \\ &= da_\ell\left(u - \frac{\ell-1}{d}\right) + a_{\ell-1} + a_{\ell-2} + \ldots + da_1 \end{aligned}$$

where

$$a_1 = \frac{1}{d(1+\frac{1}{d-1})^{d-1} - (d-1)} \quad \text{and} \quad a_\ell = a_1\left(1 + \frac{1}{d-1}\right)^{\ell-1} \ \forall\ 2 \le \ell \le d.$$

Informally, f_d is linear with coefficient da_ℓ on every interval $\left[\frac{\ell-1}{d}, \frac{\ell}{d}\right)$ for $1 \le \ell \le d$. Note that the maximum derivative of f_d is $1/C(d)$, where

$$C(d) = \frac{d\cdot(1+\frac{1}{d-1})^{d-1} - (d-1)}{d\cdot(1+\frac{1}{d-1})^{d-1}} = 1 - \frac{d-1}{d\cdot(1+\frac{1}{d-1})^{d-1}} \overset{d\to\infty}{\longrightarrow} \frac{e-1}{e}.$$

We denote the primal and dual objective values with the symbols X and Y respectively. Similarly, we refer to the changes of the objective values after the assignment of an item with ΔX and ΔY.

At each step of the original algorithm of [3], the following holds:

$$\Delta Y \leq \frac{1}{C(d)} \ \Delta X$$

Now we establish a similar bound for the new algorithm.

Lemma 4.2 (Robustness)**.** *At each step of the new algorithm the following holds:*

$$\Delta Y \leq \frac{1}{C(d)} + (1-\eta)(1-f_d(\eta)) \ \Delta X$$

When d is large enough, $f_d(\eta) \approx 1 + \frac{e(e^{\eta-1}-1)}{e-1}$ and we approximately get

$$\Delta Y \leq \left(\frac{e}{e-1} + \frac{(1-\eta)(1-e^{\eta-1})e}{e-1}\right) \ \Delta X$$

Proof. We fix an arbitrary item j and bound the ratio of increase of the primal and the dual objective values caused by the arrival of item j. We set the dual variables as

$$\forall \ i \qquad y_i = f_d\left(\frac{\sum_j b_j x_{ij}}{B_i}\right)$$

and

$$z_j = \left(1 - f_d\left(\min_{i \in S_j} \ \{\ell_i\}\right)\right) \cdot b_j$$

where ℓ_i is the level of buyer i at the end of the algorithm's execution. By the definition of the dual variables

$$\forall \ i \in S_j \qquad y_i \geq f_d\left(\min_{i' \in S_j} \ \{\ell_{i'}\}\right).$$

Therefore, $b_j \ y_i + z_j \geq b_j$ for every $i \in S_j$, that indicates that the dual variables are feasible.

Let's assume that item j is not sold completely during the execution of the algorithm. This means that all interested buyers in S_j exhausted their budget. Hence, by the definition of the dual variables, $\forall \ i \in S_j : y_i = 1$ and $z_j = 0$. The rate of change of the dual objective value related only to the y-variables is at most:

$$B_i \ \frac{\partial f_d}{\partial x_{ij}} \leq B_i \cdot \frac{b_j}{B_i} \cdot f'_d\left(\frac{\sum_j b_j x_{ij}}{B_i}\right) \leq \frac{b_j}{C(d)}$$

where $1/C(d)$ is the maximum derivative of f_d. Besides, the increasing rate of the primal objective value is b_j. Hence, the primal change is at least $1/C(d)$ that of the dual. In the remaining section of the proof, we assume that the algorithm sold all items completely. Note that z_j can be written as

$$z_j = \int_0^1 \left(1 - f_d\left(\min_{i \in S_j}\{\ell_i\}\right)\right) b_j \ dx.$$

In other words, one can imagine that during the allocation of item j, z_j is increasing at a rate of $(1 - f_d(\min_{i \in S_j} \ \{\ell_i\})) \ b_j$.

There are three stages in the algorithm. In Stages 1 and 3, the algorithm always allocates equally the fractions of item j to the buyers in S_j on the lowest level (buyers who spent the least fraction of their budget). The authors of [3] showed that during that allocation, the increasing rate of the dual is at most $1/C(d)$ that of the primal. Our proof is similar to [3], and we present it in Lemma 4.3 for completeness.

We are now interested in the allocation during Stage 2 of the algorithm. In this stage, the algorithm allocates a part of item j only to the predicted buyer i^*. The increasing rate of the dual objective w.r.t y_{i^*} is

$$B_{i^*} \frac{\partial f_d}{\partial x_{i^*j}} \leq B_{i^*} \cdot \frac{b_j}{B_{i^*}} \cdot f'_d \left(\frac{\sum_j b_j x_{i^*j}}{B_{i^*}} \right) \leq \frac{b_j}{C(d)}$$

since $f'_d(u) \leq 1/C(d)$ for all $0 \leq u \leq 1$. Note that every buyer $i \in S_j$ has spent at least η fraction of its budget before this stage, so $z_j \leq (1 - f_d(\eta))\ b_j$. Therefore, the total increasing rate of the dual in this step is at most

$$B_{i^*} \frac{\partial f_d}{\partial x_{i^*j}} + \frac{\partial z_j}{\partial x_{i^*j}} \leq \frac{b_j}{C(d)} + (1 - f_d(\eta))\ b_j.$$

Combining all the cases, the increasing rate of the dual is at most

$$\begin{cases} \frac{b_j}{C(d)} + (1 - f_d(\eta))\ b_j & \text{during Stage 2,} \\ \frac{b_j}{C(d)} & \text{during Stages 1 and 3.} \end{cases}$$

Observe that the algorithm allocates at most $(1 - \eta)$ fraction of item j during Stage 2. Therefore, the total increase of the dual due to the arrival of j is at most:

$$\int_0^{(1-\eta)} \left(\frac{b_j}{C(d)} + (1 - f_d(\eta))\ b_j \right) dx + \int_0^{\eta} \left(\frac{b_j}{C(d)} \right) dx = \left(\frac{1}{C(d)} + (1-\eta)(1 - f_d(\eta)) \right) b_j.$$

Besides, the increase of the primal is b_j. Hence, we can deduce that the robustness of the algorithm is

$$\frac{1}{\left(\frac{1}{C(d)} + (1-\eta)(1 - f_d(\eta)) \right)}.$$

Finally, we compute and estimate the value of $f_d(\eta)$ for large value of d. We have

$$\begin{aligned} f_d(\eta) &= f_d\left(\frac{\lfloor \eta \cdot d \rfloor}{d} \right) \\ &= a_1 \left(d \left(1 + \frac{1}{d-1} \right)^{\lfloor \eta \cdot d \rfloor - 1} - (d-1) \right) \\ &= \frac{d(1 + \frac{1}{d-1})^{\lfloor \eta \cdot d \rfloor - 1} - (d-1)}{d(1 + \frac{1}{d-1})^{d-1} - (d-1)} \\ &= 1 + \frac{d(1 + \frac{1}{d-1})^{\lfloor \eta \cdot d \rfloor - 1} - d(1 + \frac{1}{d-1})^{d-1}}{d(1 + \frac{1}{d-1})^{d-1} - (d-1)} \end{aligned}$$

$$= 1 + \left(\frac{d(1+\frac{1}{d-1})^{\lfloor \eta \cdot d \rfloor - 1}}{d(1+\frac{1}{d-1})^{d-1}} - 1 \right) \frac{1}{1 - \frac{(d-1)}{d(1+\frac{1}{d-1})^{d-1}}}$$

$$= 1 + \frac{1}{C(d)} \left(\left(1 + \frac{1}{d-1}\right)^{\lfloor \eta \cdot d \rfloor - d} - 1 \right) \overset{d \to \infty}{\longrightarrow} 1 + \frac{e(e^{\eta - 1} - 1)}{e - 1}$$

which concludes the proof. □

Remark. As we mentioned in Subsection 3.3 regarding the component observations, the assignment of Stage 1 in the meta-algorithm mitigates the risk of the prediction assignments. In the proof of Lemma 4.2 we relied on the fact that every interested buyer spent at least η fraction of its budget to set the z_j dual variable. Without this assumption, the bound on this variable is

$$z_j \leq (1 - f_d(0))\ b_j = (1 - 0)\ b_j$$

which decreases the meta-algorithm's robustness to

$$\frac{1}{\left(\frac{1}{C(d)} + (1 - \eta)\right)}.$$

We can accept this decrease if the algorithm has complete trust in the prediction. However in a realistic setting this assumption is too strong, therefore Stage 1 is essential.

Lemma 4.3. *We assume that j is completely sold by the algorithm. Then, at all times during the allocations of Stage 1 and 3 of the new algorithm, the increasing rate of the dual objective value is at most $1/C(d)$ that of the primal.*

Proof. The proof follows the one in Buchbinder and Naor [3, Theorem 13.1]. Recall that item j was completely sold. We consider two cases.

Case 1: The highest level item j was sold at is ℓ, and at the end of the algorithm, all buyers in S_j spent at least $\frac{\ell+1}{d}$ fraction of their budget. By definition, the dual variables are bounded with:

$$z_j \leq \left(1 - f_d\left(\frac{\ell+1}{d}\right)\right)\ b_j$$

Recall that as j is completely sold, i.e., $\sum_i x_{ij} = 1$. We can consider this as the increasing rate of z_j at any time during the allocation of j is at most $(1 - f_d(\frac{\ell+1}{d}))\ b_j$. Besides, as the highest level item j was sold at is ℓ, the rate of change of the dual objective value due to the change in y_i is:

$$B_i \frac{\partial f_d}{\partial x_{ij}} \leq B_i \cdot \frac{b_j}{B_i} \cdot d \cdot a_{\ell+1} \leq b_j \cdot d \cdot a_{\ell+1}$$

Therefore, the total change of the dual is at most

$$
\begin{aligned}
& b_j d a_{\ell+1} + b_j \left(1 - f_d\left(\frac{\ell+1}{d}\right)\right) \\
&= b_j d a_1 \left(1 + \frac{1}{d-1}\right)^{\ell} + b_j \left(1 - a_1\left(d\left(1 + \frac{1}{d-1}\right)^{\ell} - (d-1)\right)\right) \\
&= b_j(1 + a_1(d-1)) = \frac{b_j}{C(d)} \qquad (3)
\end{aligned}
$$

Case 2: The highest level item j was sold at is ℓ, and at the end of the algorithm, at least one buyer in S_j spent less than $\frac{\ell+1}{d}$ fraction of its budget. In this case we need to set $z_j = (1 - f_d(\ell/d))\ b_j$ to satisfy the dual constraint. During any (short) period of length Δt, the increase of $\sum_i B_i\ y_i$ over i in the highest level is at most:

$$
b_j\ a_{\ell+1}\ d\ \frac{d-1}{d}\ \Delta t = b_j\ a_{\ell+1}\ (d-1)\ \Delta t,
$$

where the fraction $(d-1)/d$ is there because there are at most $(d-1)$ buyers in the highest level. The increase of $\sum_i B_i\ y_i$ over i in the lower levels is at most $b_j\ d\ a_\ell\ (1 - \Delta t)$. By definition, $a_{\ell+1} = a_\ell\ (1 + 1/(d-1))$ and $a_\ell = ((d-1)/d)\ a_{\ell+1}$. Therefore, by adding things together, the increasing rate of $\sum_i B_i\ y_i$ is at most $b_j\ (d-1)\ a_{\ell+1}$. Hence, the increasing rate of the dual is at most:

$$
\begin{aligned}
& b_j\ (d-1)\ a_{\ell+1} + b_j \left(1 - f_d\left(\frac{\ell}{d}\right)\right) \\
&= b_j\ (d-1)\ a_{\ell+1} + b_j \left(1 - f_d\left(\frac{\ell+1}{d}\right) + a_{\ell+1}\right) \\
&= b_j\ d\ a_{\ell+1} + b_j \left(1 - f_d\left(\frac{\ell+1}{d}\right)\right) = \frac{b_j}{C(d)}
\end{aligned}
$$

where the first equality holds since $f_d\big(\frac{\ell+1}{d}\big) = f_d\big(\frac{\ell}{d}\big) + a_{\ell+1}$ and the last equality follows from Equation (3). In both cases the increasing rate of the dual is at most $b_j/C(d)$, whereas that of the primal is b_j. The lemma follows. □

Theorem 4.4. *The new algorithm is $(1-\eta)$-consistent with the prediction and*

$$
1/\left(\frac{1}{C(d)} + (1-\eta)(1 - f_d(\eta))\right)\text{-robust}
$$

with the optimal offline solution. When d is large we approximately get

$$
1/\left(\frac{e}{e-1} + \frac{(1-\eta)(1 - e^{\eta-1})\ e}{e-1}\right)\text{-robustness.}
$$

Proof. Follows from Lemma 4.1 and Lemma 4.2. □

5 Experiments

5.1 Implementation

We implemented the new meta-algorithm to have an experimental evaluation of the algorithm's performance. The code is available on GitHub[1]. Python ($\geq$ 3.9.1) and Gurobi ($\geq$ 9.5.0) are necessary to execute the program. We note that Gurobi provides a free academic licence. Therefore, anyone can execute the code without payment.

Input. There are two ways to formulate an instance for the program, namely, through manual and random inputs. We list manual inputs in the `src/manual_inputs.py` file, while random inputs in the `src/configuration.py` file. Random inputs require a configuration that sets the range for the values of the instance, for example, the minimum and maximum budget of a buyer. We draw uniformly at random the values within the configuration ranges for the variables of the instance. The input generation for random inputs is implemented in the `src/input_generation.py` file. There are three base structures in the implementation, namely, the classes `Buyer`, `Item` and `ProblemInput`. These are written in the `src/input.py` file.

Execution. The entry point of the program is the `main.py` file. Following the input generation, the program uses Gurobi to solve the linear program of the bounded allocation problem. We rely on the linear program formulation to calculate the instances' optimal offline objective value. To fasten the execution time, we store the solutions of the linear programs of each instance in the `cache` folder. The linear program solving related logic is written inside the `src/lp_solver.py` file. Afterwards, the program runs several experiments using the meta-algorithm's implementation. Finally, the program saves the results in the `output` folder and displays a plot to visualize the competitive ratio of the algorithm. The implementation of the meta-algorithm is inside the `src/bounded_allocation_solver.py` file.

Parameters. We can set execution related parameters through command line arguments. There are three parameters that influence the number of experiments the program runs. In case we execute the program with randomly generated instances, by setting the parameter `-r <integer>`, we can obtain a result from the average of the executions. This means, that we can fix the configuration of the random instance and generate random inputs with several different random seeds. Additionally, by setting the parameter `-e <float 1> <float 2> ... <float k>`, we can run each input with several different prediction error rates. Finally, we can set the granularity with which the program explores η values in the range of $[0, 1]$. By setting the parameter `-n <integer>`, the program experiments with η values from $^0/_n$ to $^n/_n$. Figure 10 below shows the nested relation between these three parameters.

Predictions. We create predictions for the experimental instances by introducing perturbations to the optimal offline integral solution. The perturbation is skipped when $|S_j| = 1$. When $|S_j \setminus \{i^*\}| > 1$ we choose uniformly at random a remaining buyer to replace the optimal buyer. The corresponding logic is implemented in `src/prediction.py` file.

[1]https://github.com/314szke/bounded_allocation

5.2 Instances

We present here four experiments. Table 2 below summarizes the configurations of the instances. We show a mini-figure for each instance on Figures 12-15, displaying the competitive ratio of the algorithms, that we calculate by dividing the objective value of the algorithm, ALGO(I), with the objective value of the optimal fractional offline solution, OPT(I). The x-axis corresponds to the ratio with which the algorithm considers the prediction. We indicate with $\eta = 0$ no doubt and $\eta = 1$ complete doubt in the prediction. Therefore, the left hand side of the figures correspond to a prediction dominated, while the right hand side to a water-filling dominated meta-algorithm. The different colors on the figure correspond to different prediction error rates.

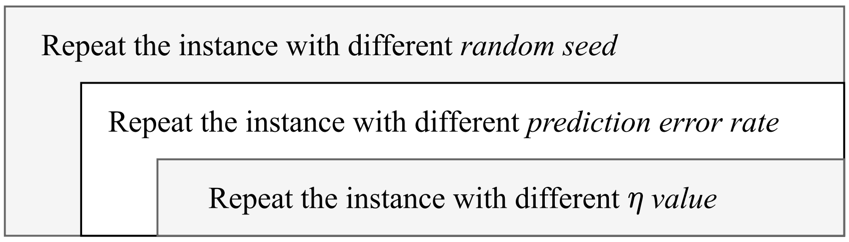
Repeat the instance with different *random seed*
Repeat the instance with different *prediction error rate*
Repeat the instance with different *η value*

Figure 10: Structure of the experiment iterations

Instance 1 corresponds to one of the pathological inputs for the water-filling algorithm. We created this instance manually to observe the behaviour of the meta-algorithm when the nested online algorithm of [3] performs poorly. In the scenario of Instance 1, each item j is connected to each buyer i that has an index $i \geq j$. The structure of Instance 1 is visible on Figure 11. The optimal solution is to sell each item j to buyer i, where $i = j$, while the water-filling algorithm attempts to allocate each item equally.

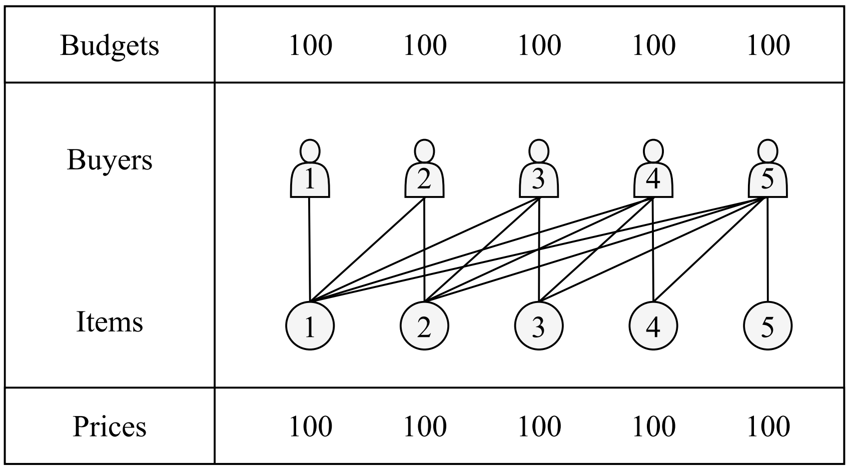

Figure 11: Pathological example for the water-filling algorithm

Instances 2-4 are randomly generated based on their corresponding configuration visible in Table 2. We executed each instance 20 times. The lines on Figures 12-15 correspond to the average of these executions and the colored areas correspond to the 95% confidence intervals. Instance 2 and Instance 3 mimic real-life instances, where the general expectation is to have a small bound d on the interested buyers. The average item price over the average budget value across the executions was 6.36% for Instance 2 and 0.98% for Instance 3. Finally, Instance 4 shows an example where the integral solution (and therefore the prediction) is not optimal. Instance 4 has a large bound on the number of interested buyers and the items' prices vary greatly. Tables 3-6 show metrics related to the instances and Figures 17-20 the observed integrality gaps.

5.3 Evaluation

Instance 1. Looking at Figure 12 we can observe that the meta-algorithm performs identically with several prediction error rates. For the values 0.0 and 0.1 the competitive ratio increases as η decreases. Similarly, for the values $0.2, 0.3$, and 0.4 the performance of the meta-algorithm follows the lower line. We can conclude that the meta-algorithm outperforms the traditional water-filling algorithm whenever $\eta < 0.8$ even with high prediction error rates.

Instance 2. The second instance is a good example to show the robustness of the meta-algorithm. Looking at the y-axis of Figure 13 we can notice that the meta-algorithm performs close to optimality with every observed prediction error rate. On Figure 18 we can observe that the integrality gap is zero for every random iteration of Instance 2.

Instance 3. Out of the four experimental instances, the third instance is the closest to represent a real-life scenario. There are significantly more items than buyers in this setting and the items have small prices compared to the buyers' budgets. Additionally, the bound on the number of interested buyers is small. Figure 14 presents the results of this instance. We can observe that with reasonable error rates the meta-algorithm can improve the performance of the water-filling algorithm with around 1% if we consider $\eta \in [0.2,\ 0.6]$. Even though this performance increase does not seem significant at first, we note that the bounded allocation problem has practical applications in use cases such as advertising, where 1% of increased revenue is non-negligible. Figure 19 shows that there is a minor integrality gap in around half of the random iterations.

Instance 4. Finally, Instance 4 rerpesents a difficult case for the meta-algorithm due to the large integrality gap. Figure 20 shows that on average the random iterations have around 17% integrality gap. We can see on Figure 15 that even with perfect predictions the performance of the meta-algorithm degrades with decreasing η values. The integrality gap is so large that regardless the prediction error rate, the competitive ratio decreases with the same tendency.

Overall. The experiments confirm the water-filling algorithm's benefit from the prediction information. We showed that the meta-algorithm has improved performance on the pathological input for the original water-filling algorithm even with high prediction error rates. Besides, the meta-algorithm demonstrated firm robustness against elevated prediction error rates in the second instance, where the water-filling algorithm performed close to optimality. Furthermore, we observed improved performance on the third instance, which represents a close to real-life use case. In the fourth instance, we can remark the drawback of the predictions. When the integral solution of the linear programs is not optimal or even far from optimal, the prediction can significantly misguide the meta-algorithm.

Name	Type	Buyers	Items	$d = \max\{\lvert S_j\rvert\}$	Budget range	Item price range
Instance 1	manual	5	5	5	100 – 100	100 – 100
Instance 2	random	100	1.000	5	10 – 100	0.1 – 8
Instance 3	random	100	10.000	3	10 – 1.000	1 – 10
Instance 4	random	80	80	40	10 – 100	10 – 100

Table 2: Properties of the experiment instances

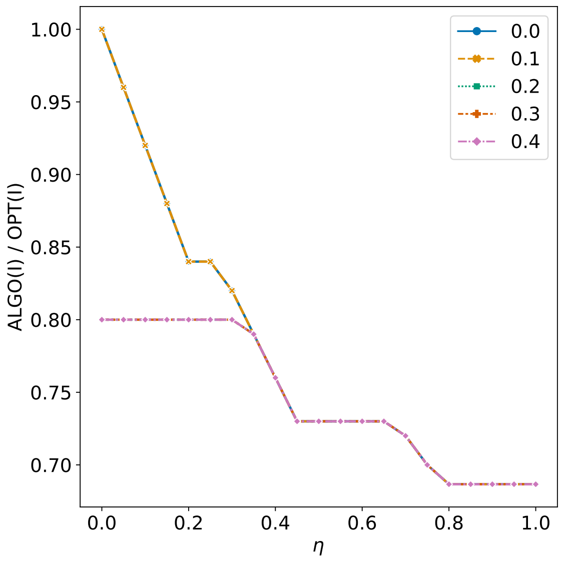

Figure 12: Instance 1

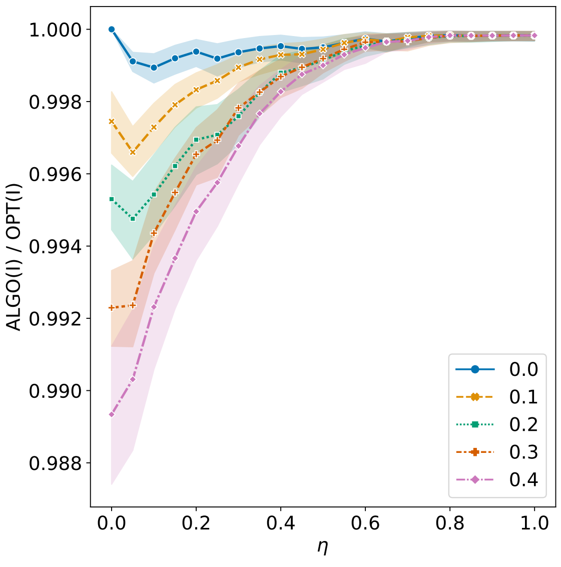

Figure 13: Instance 2

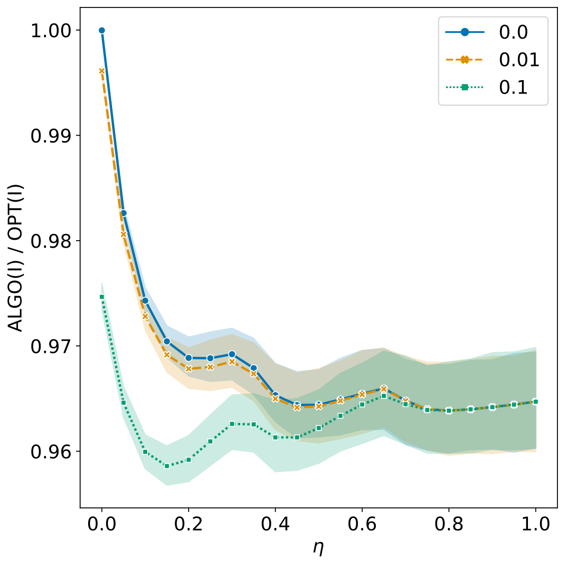

Figure 14: Instance 3

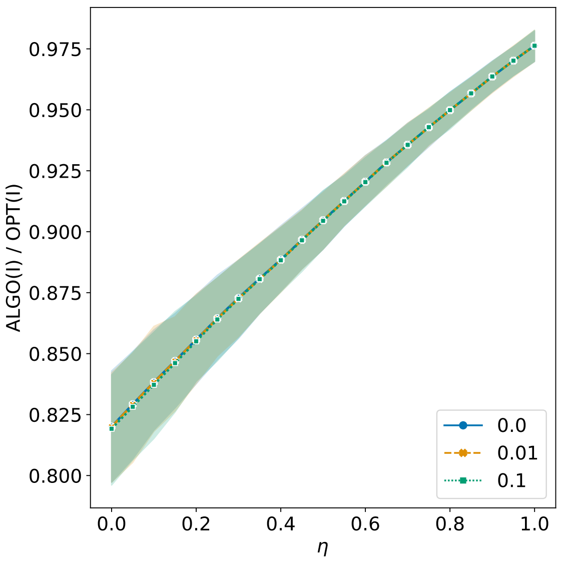

Figure 15: Instance 4

Figure 16: Observed competitive ratios

Metrics	Minimum	Maximum	Average
Budget	100	100	100
Item Price	100	100	100
Number of Buyers per Item	1	5	3
Number of Items per Buyer	1	5	3
Expected Expenses	100	500	300
Integrality gap	0.00	0.00	0.00

Table 3: Intentional metrics of Instance 1

Metrics	Minimum	Maximum	Average
Budget	10.70	99.55	54.25
Item Price	0.00	7.00	3.45
Number of Buyers per Item	2.00	5.00	3.00
Number of Items per Buyer	21.45	50.20	34.55
Expected Expenses	77.80	212.25	139.40
Integrality gap	0.00	0.00	0.00

Table 4: Observed metrics of Instance 2

Metrics	Minimum	Maximum	Average
Budget	18.65	990.70	501.06
Item Price	1.00	9.00	4.95
Number of Buyers per Item	2.00	3.00	2.00
Number of Items per Buyer	211.25	289.65	249.30
Expected Expenses	1046.65	1492.55	1262.80
Integrality gap	0.00	0.01	0.003

Table 5: Observed metrics of Instance 3

Metrics	Minimum	Maximum	Average
Budget	10.95	98.95	55.30
Item Price	10.00	98.00	49.9
Number of Buyers per Item	1.15	39.80	19.7
Number of Items per Buyer	11.75	29.35	19.70
Expected Expenses	570.00	1598.75	1044.20
Integrality gap	6.47	28.56	17.99

Table 6: Observed metrics of Instance 4

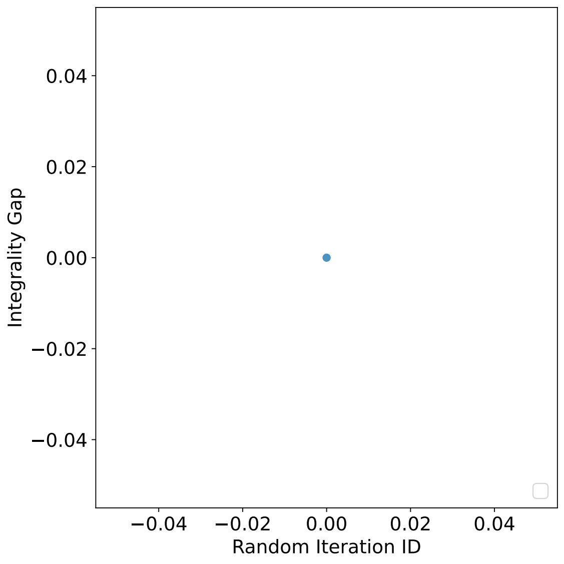

Figure 17: Instance 1

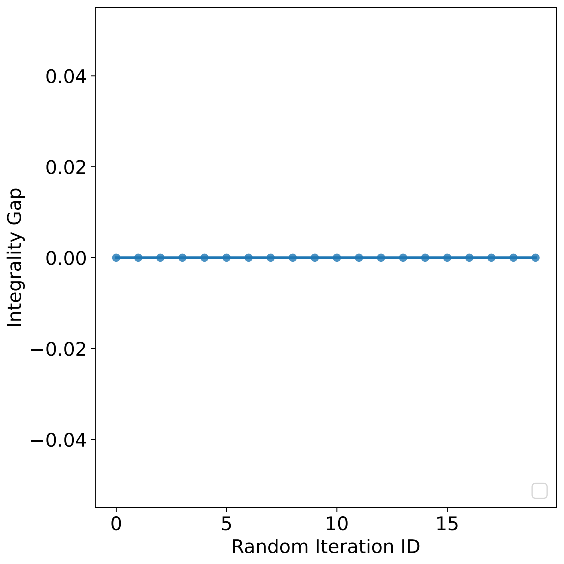

Figure 18: Instance 2

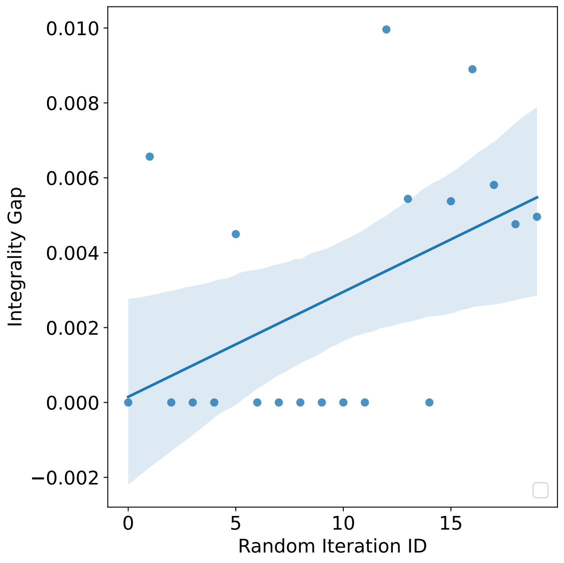

Figure 19: Instance 3

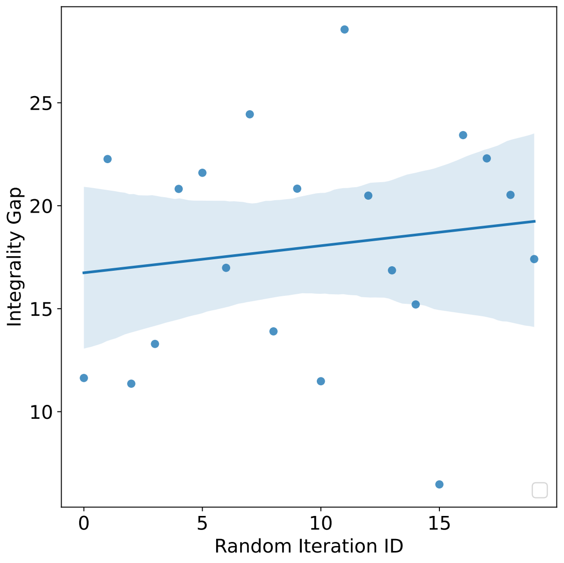

Figure 20: Instance 4

Figure 21: Integrality gaps over the random interations

6 Summary

6.1 Conclusion

Learning augmented algorithms is an active topic of research with considerable open challenges. During the short time of the internship, we developed an algorithm that includes prediction information for the bounded allocation problem with proven performance guarantees (see Theorem 4.4). The consistency of the meta-algorithm is competitive with the consistency guarantee of similar problems in the literature, for example, as in [1] and [11].

Unfortunately, due to the lack of time, we did not finalize a proof to obtain the optimality bound (or lower bound) of the problem to evaluate the quality of the algorithm's robustness (competitive ratio). Nevertheless, the experimental results seem promising. Additionally, the program we implemented to experiment with the algorithm can aid future work. The modular structure of the code can provide the building blocks for new algorithms, and the experiments can offer insight into further performance improvements.

6.2 Conference paper submission

Coupled with results obtained outside of the internship, we submitted a paper to the European Symposium on Algorithms (the description is available here: ESA[2]). The paper incorporates the algorithm, related proofs and experiments we presented here. As of the time of writing, we did not receive an answer regarding the acceptance.

6.3 Future work

The presented work is still in its early stage, and we plan to continue the research during a PhD thesis. As the experimental results showed, it might be beneficial to consider more detailed prediction information. For example, the prediction can provide a set of weights to divide the item among the interested buyers. However, augmenting the complexity of the prediction and how the meta-algorithm combines the water-filling algorithm with the prediction may pose significant difficulties in proving the robustness. Moreover, our proposed meta-algorithm has a fixed policy in the sense that the value of η is fixed before the execution and does not change over time. The meta-algorithm could keep track of the accuracy of the prediction and decide what value of η is the best in the given state of the instance. Regardless, it is vital to find the right balance between the algorithm's complexity and the performance metric proof's complexity.

The uncertainty in the bounded allocation problem revolves around the items. In our setting, we assumed that an oracle could guide the decision making based on previously gathered data. As we mentioned in the related work subsection in the introduction, there are several possibilities to explore controllable parts of the uncertainty. In some use cases, we could assume that there are reoccurring items that we can group into categories. Based on previous observations, it may be possible to determine the frequency of occurrence of these items and provide a probability distribution over the categories to the meta-algorithm.

[2]https://algo2022.eu/esa/#cfp

To go further, it may be fruitful to consider other algorithm design techniques, which do not rely on the primal-dual method, for example as in [10]. Analysing different predictions and determining error metrics to evaluate them can provide valuable insights on the problem, regardless the algorithm design technique we use.

Another substantial milestone in our research is evaluating the meta-algorithm's performance in a real-life use case. A concrete example might also provide insights into the explorable parts of the problem's uncertainty.

6.4 Acknowledgements

I would like to express my gratitude towards MIAI for providing the financial aide for my internship. I am thankful for the warm welcome of the DATAMOVE team in LIG, and I hope that I can continue to work with them. Lastly, I truly appreciate the guidance and patience of my supervisors: Thang Nguyen Kim and Denis Trystram. I am looking forward to our work together during the PhD thesis.

References

[1] Etienne Bamas, Andreas Maggiori, and Ola Svensson. “The Primal-Dual method for Learning Augmented Algorithms”. In: *Proc. 34th Conference on Neural Information Processing Systems*. 2020.

[2] *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021. DOI: 10.1017/9781108637435.

[3] Niv Buchbinder and Joseph (Seffi) Naor. “The Design of Competitive Online Algorithms via a Primal: Dual Approach”. In: *Found. Trends Theor. Comput. Sci.* 3.2–3 (Feb. 2009), pp. 93–263. ISSN: 1551-305X. DOI: 10.1561/0400000024. URL: https://doi.org/10.1561/0400000024.

[4] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. “Online primal-dual algorithms for maximizing ad-auctions revenue”. In: *European Symposium on Algorithms*. 2007, pp. 253–264.

[5] Hossein Esfandiari, Nitish Korula, and Vahab S. Mirrokni. “Online Allocation with Traffic Spikes: Mixing Adversarial and Stochastic Models”. In: *CoRR* abs/1711.05764 (2017). arXiv: 1711.05764. URL: http://arxiv.org/abs/1711.05764.

[6] Morteza Ghobakhloo. “Industry 4.0, digitization, and opportunities for sustainability”. In: *Journal of Cleaner Production* 252 (2020), p. 119869. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.119869. URL: https://www.sciencedirect.com/science/article/pii/S0959652619347390.

[7] Bobby Kleinberg. “Online bipartite matching algorithms”. In: *Cornell University* CS 6820: Algorithms (2012). URL: https://www.cs.cornell.edu/courses/cs6820/2012fa/handouts/online-matching.pdf.

[8] Ravi Kumar et al. “Semi-Online Bipartite Matching”. In: *CoRR* abs/1812.00134 (2018). arXiv: 1812.00134. URL: http://arxiv.org/abs/1812.00134.

[9] Benedikt Latos et al. “Complexity Drivers in Digitalized Work Systems: Implications for Cooperative Forms of Work”. In: *Advances in Science, Technology and Engineering Systems Journal* 3 (Sept. 2018), pp. 166–185. DOI: 10.25046/aj030522.

[10] Alexander Lindermayr and Nicole Megow. *Permutation Predictions for Non-Clairvoyant Scheduling*. 2022. DOI: 10.48550/ARXIV.2202.10199. URL: https://arxiv.org/abs/2202.10199.

[11] Kim Thang Nguyen and Christoph Durr. “Online Primal-Dual Algorithms with Predictions for Packing Problems”. In: *CoRR* abs/2110.00391 (2021). arXiv: 2110.00391. URL: https://arxiv.org/abs/2110.00391.

[12] The Shift Project. “Lean ICT: Towards digital soberity”. In: *The Shift Project* (2019). URL: https://theshiftproject.org/wp-content/uploads/2019/03/Lean-ICT-Report_The-Shift-Project_2019.pdf.

[13] W.E. Walker et al. “Defining Uncertainty: A Conceptual Basis for Uncertainty Management in Model-Based Decision Support”. In: *Integrated Assessment* 4.1 (2003), pp. 5–17. DOI: 10.1076/iaij.4.1.5.16466. eprint: https://doi.org/10.1076/iaij.4.1.5.16466. URL: https://doi.org/10.1076/iaij.4.1.5.16466.