ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

PROJECT: ALGOLINC – Algorithmic Learning Theory and Incentives: Synergies in Optimization and Mechanism Design

Project ID: 15877

Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN "BASIC RESEARCH FINANCING" (Horizontal support for all Sciences)

WP5: Data-driven and Learning-augmented Mechanism Design

Deliverable D.5.2: Technical report (including research submitted to relevant conferences or journals) on learning-augmented mechanisms

Prepared by the team members:

Georgios Amanatidis, Evangelos Markakis, Panagiotis Patsilinakos, Panagiotis Tsamopoulos

ΕΛΙΔΕΚ

Ελληνικό Ίδρυμα Έρευνας & Καινοτομίας

Deliverable D.5.2: Technical report (including research submitted to relevant conferences or journals) on learning-augmented mechanisms

1 Overview of Publications

This deliverable concerns WP5, and was completed by Month M22, with a 2 month delay compared to the initial expected delivery date. This was caused by the fact that two post-doctoral researchers left the team and consequently, we had to initiate the process of hiring new members of the research team as a replacement.

The goal of this report is to provide an overview of our progress regarding the completion of WP5 along with the research papers that were written by our team on the relevant topics under consideration.

The output of our research within WP5 consists of 2 published papers in peer-reviewed conferences (one at ICML 25 and one at SAGT 25), and one paper that will soon be submitted to a journal. The relevant information regarding these papers is as follows.

- Evripidis Bampis, Bruno Escoffier, Dimitris Fotakis, Panagiotis Patsilinakos, Michalis Xefteris. Polynomial Time Learning Augmented Algorithms for NP-hard Permutation Problems. In *Proceedings of the Forty-second International Conference on Machine Learning*, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025.

- Georgios Amanatidis, Evangelos Markakis, Christodoulos Santorinaios, Guido Schaefer, Panagiotis Tsamopoulos, Artem Tsikiridis. Online Budget-Feasible Mechanism Design with Predictions. In *Proceedings of the 18th International Symposium on Algorithmic Game Theory*, SAGT 2025, pages 402-421, Bath, UK, September 2-5, 2025.

- Georgios Amanatidis, Alexandros Lolos, Evangelos Markakis, Victor Turmel. Online Fair Division for Personalized 2-Value Instances. To be submitted to the *Journal of Artificial Intelligence Research*, JAIR, 2025. Available at `https://arxiv.org/abs/2505.22174`.

The above three papers can be found at the Appendix of this deliverable.

2 Introduction

We briefly recall first the research topics under consideration. In the recent years there has been a growing interest in learning-oriented approaches in algorithm design. This is relevant for various problems in combinatorial optimization, for resource allocation problems, as well as for procurement auctions for assigning tasks to competing providers. What can be common in all these problems is the uncertainty that could arise in the online

version where e.g., the input is revealed sequentially and decisions need to be made without knowing the future. Another source of uncertainty, especially for algorithmic problems arising within game theoretic environments, comes from the fact that the preferences of the involved agents may not be fully known e.g., in resource allocation problems or in auctions, the algorithm designer may need to elicit such preferences from startegic agents who may have incentives to misreport.

One way for tackling such algorithmic questions is to attempt to go beyond the typical worst-case analysis in algorithm design. In order to follow such an approach, one needs to make informational assumptions that the algorithm designer could use and exploit. This has created a new research agenda recently, and is very meaningful especially when these assumptions can be motivated by the availability of past data or statistics. One of the main directions along this approach is to assume that the algorithm designer has at her disposal some prediction information regarding relevant parameters of the problem or even regarding the optimal solution itself. E.g. in problems related to online versions of vehicle routing or TSP, the prediction could involve the geographic location of requests. In auctions, the predictions could involve the preferences of the competing bidders or the cost of an optimal procured solution. Since the prediction may not always be correct, the goal would be to attain the "best of both worlds", by designing algorithms that have a good performance when the prediction is precise, but also achieve an approximately optimal worst-case guarantee when the prediction fails. During the last years, this is referred to as the "learning-augmented" paradigm[1] and has attracted significant attention ever since the work of [4] that brought it forward as a promising approach.

Another learning-oriented direction that is meaningful in online versions of such problems to assume that we have access to correct "look-ahead" predictions. E.g. in settings where we need to serve in an online fashion incoming requests, one could assume that we have access to the information relevant to the next k requests, for some small parameter k. In the same spirit, in online allocation mechanisms where goods or tasks arrive online, and need to be irrevocably allocated to competing agents, one could have access to the preferences of the agents for the next k incoming goods. Even when k is a very small constant, such as $k = 1$ or $k = 2$, this could still give rise to positive results for some problems, and can be seen as a way to alleviate partially the existing uncertainty of the future input. A further generalization of this is to envision a *query model*, where the central entity can make queries on the agents' preferences or on other parameters of the underlying problem. Queries could be of cardinal or ordinal nature, e.g., "what is your top preference among the available goods of the auction"? Such queries come at a cost (cost of information acquisition). The relevant questions therefore concern how to obtain solutions that make a polynomial number of queries, or even further the minimum possible number of queries.

3 Learning-augmented mechanisms for procurement auctions

Procurement auctions, also known as reverse auctions, are mechanisms for assigning a set of projects or tasks to a set of candidate providers. In the more traditional applications of reverse auctions, the auctioneer may be a governmental organization (e.g., for assigning a public project to a construction firm), or a company interested in outsourcing tasks, such

[1] For an overview of recent publications on learning-augmented algorithms, we refer to [3] and especially to the site `https://algorithms-with-predictions.github.io/`.

as the supply of logistics or retail services. In recent years, the explosion and monetization of the Internet, have created even further applications and platforms where even a single individual can conduct online assignments/auctions. Two such prominent families of examples include platforms for online labour markets, such as Upwork or Freelancer, and platforms for crowdsourcing systems, where the posted tasks can range from very elementary assignments as image labeling, all the way to more specialized services like translating documents. Participatory crowdsensing also falls under this framework, where some entities are interested in collecting sensing information from smartphones or other devices.

The emergence of such online markets has caught the increasing attention of the research community, which has focused on properly formulating and addressing relevant research questions. One of the popular models in the literature for capturing procurement auctions is that of *budget-feasible mechanism design*, introduced by [5]. This model concerns a scenario where an auctioneer with a strict budget constraint seeks to purchase goods or services from a set of strategic agents, who may misreport their cost to their advantage for obtaining higher payments. Under these considerations, a natural goal for the auctioneer is to come up with a truthful mechanism for hiring a subset of the agents that maximizes the procured value (measured by the valuation function of the auctioneer), and such that the total payments to the agents respect the budget limitation. Given that even the non-strategic versions of such budget-constrained problems tend to be NP-hard, the main focus is on providing budget-feasible mechanisms that achieve approximation guarantees on the auctioneer's optimal potential value.

The focus of our work is to study budget-feasible mechanisms under the framework of learning-augmented mechanism design, as briefly described in Section 2. The main goal is to leverage a (possibly erroneous) prediction to improve worst-case performance guarantees when the prediction is accurate (referred to as *consistency*), while also providing a performance guarantee when it is not (referred to as *robustness*), without knowing the prediction accuracy in advance. These notions are defined in Section 3.1. Recently, there have been a few other settings with strategic agents, that have been considered through this approach. Our work has also been motivated by such attempts.

3.1 Model, notation and problem definitions

We explored the online setting of mechanism design with predictions, focusing on a procurement auction involving a single auctioneer and multiple agents. The auctioneer has a valuation function $v : 2^N \to \mathbb{Q}_{\geq 0}$ and a budget $B > 0$. We use $N = [n] = \{1, 2, \ldots, n\}$ to denote the set of n agents. Each agent i has a *private* cost $c_i > 0$, namely the cost of getting hired by the auctioneer. For $S \subseteq N$, $v(S)$ represents the value the auctioneer derives from selecting the set of agents S; for singletons, we write $v(i)$ instead of $v(\{i\})$. The algorithmic goal in all the problems we study is to select a set S that maximizes $v(S)$ subject to the constraint $\sum_{i \in S} c_i \leq B$. A function v is non-decreasing (referred to as *monotone* herein), if $v(S) \leq v(T)$ for any $S \subseteq T \subseteq N$. We mostly focus on the (rather popular for this setting) class of non-negative *submodular* valuation functions, whether monotone or not, defined below.

Definition 3.1. *A valuation function $v(\cdot)$ is submodular if $v(i|S) \geq v(i|T)$ for all $S \subseteq T$ and $i \notin T$.*

Note that when $v(i|S) = v(i|\emptyset)$, for all $i \in N$ and all $S \subseteq N$, the valuation function v is additive.

An important element in our problem is the online arrival of agents. We adopt the *random order* (RO) model. In this model an adversary first chooses the costs of all the agents in N. There are n different time slots $t_1, \ldots, t_n$, with a single agent appearing in each slot according to a uniformly random permutation π, i.e., the input sequence to an *online* algorithm is $\pi(1), \pi(2), \ldots, \pi(n)$. Such an algorithm has to take its irrevocable decision for agent $\pi(i)$ (whether to hire her or not) before seeing agent $\pi(i+1)$. The length n of the input sequence may be known in advance, and we assume so in our setting. For the RO model, given the output A of an algorithm and the value of an optimal solution OPT, we define the competitive-ratio (for maximization problems) to be $\frac{OPT}{\mathbb{E}[v(A)]}$ on the adversarily-chosen (worst case) set of inputs. The expected value is over all permutations of the input.

Mechanism Design. An offline mechanism $M = (A, p)$ consists of an allocation function $A : \mathbb{R}^n_{\geq 0} \to 2^{[n]}$ and a payment rule $p : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$. Given a profile of cost declarations $\mathbf{b} \in [0, B]^n$, where each b_i is the cost *reported* by agent i (which may differ from the actual cost c_i for strategic reasons), the allocation function selects a set of hired agents $A(\mathbf{b}) \subseteq N$. The payment rule returns a profile of payments $\mathbf{p}(\mathbf{b}) = (p_1, \ldots, p_n)$, where $p_i(\mathbf{b}) \geq 0$ is the payment to agent $i \in N$. Therefore, the utility of agent $i \in N$ is $u_i(\mathbf{b}) = p_i(\mathbf{b}) - c_i$ if $i \in A(\mathbf{b})$ and $u_i(\mathbf{b}) = 0$ otherwise, with the understanding that agents not in $A(\mathbf{b})$ are not paid.

An *online* mechanism, on the other hand, decides the allocation and payment of an agent at every time step without having seen the entire input. That is, given a permutation π on N, the mechanism decides about agent $\pi(t)$ at time step $t \in [n]$, as $\pi(t)$ appears and declares her cost, i.e., at the time the mechanism has only seen $\mathbf{b}^\pi_t = (b_{\pi(1)}, \ldots, b_{\pi(t)}) \in [0, B]^t$. The mechanism, thus, maintains the pair $(A_t(\mathbf{b}^\pi_t), p_t(\mathbf{b}^\pi_t))$ of currently selected agents and their payments, which is consistent with the decisions being irrevocable.

For a *deterministic* mechanism $M = (A, p)$ we require that, for every fixed arrival order, any true cost profile $\mathbf{c}$ and any declared cost profile $\mathbf{b}$, the mechanism is:

- *budget-feasible*; the payments to the agents do not exceed the auctioneer's budget, i.e., $\sum_{i=1}^n p_i(\mathbf{b}) \leq B$.
- *individually rational*; no agent can be hired for less than they asked for, i.e., if $i \in A(\mathbf{b})$, then $p_i(\mathbf{b}) \geq b_i$.
- *truthful*; no agent $i \in N$ has an incentive to misreport their true cost c_i, i.e., $u_i(c_i, \mathbf{b}_{-i}) \geq u_i(\mathbf{b})$.

A *randomized* mechanism $M = (A, p)$ can be thought of as a probability distribution over deterministic mechanisms for every fixed arrival order. In this work, we require all our randomized mechanisms to be probability distributions over budget-feasible, individually rational, and truthful deterministic mechanisms. Randomized mechanisms with the latter property are called *universally truthful*.[2] Since all our proposed mechanisms are provably universally truthful, we can simply refer to the true cost profile $\mathbf{c}$ instead of the declared cost profile $\mathbf{b}$ when needed.

Given the above, an instance of the problem is described by the tuple $I = (N, \mathbf{c}, v, B)$. A common practice in this line of work is to measure the performance of a mechanism $M = (A, p)$, both in online variants and the offline problem, against the optimal value of the following packing problem:

[2]This is stronger than *truthfulness in expectation*, which requires that truth-telling maximizes each agent's expected utility.

$$\max\ v(S) \quad \text{s.t.} \quad \textstyle\sum_{i\in S} c_i \leq B, \quad S \subseteq N. \tag{1}$$

We use $S^*(\mathbf{c}) \subseteq N$ to refer to an optimal solution of equation 1.

Predictions. Following what we discussed in Section 2, we employ a beyond-worst-case analysis perspective to measure the performance of mechanisms. We assume that each instance $I = (N, \mathbf{c}, v, B)$ is *augmented* by a (possibly erroneous) deterministic prediction ω of $v(S^*(\mathbf{c}))$, and that a mechanism is equipped with an error *tolerance* parameter $\tau \in [0, 1)$, set by the mechanism designer, representing how much tolerance she has for the prediction's inaccuracy. We denote the augmented instance as $I^+ = (N, \mathbf{c}, v, B, \omega)$. As is standard in the literature of algorithms with predictions, we assess the performance of a mechanism with tolerance τ, $M_\tau = (A, p)$, on augmented instances based on the following two metrics:

- *Consistency:* M_τ is α-consistent if $\alpha \cdot \mathbb{E}\left[v(A(\mathbf{c}))\right] \geq v(S^*(\mathbf{c}))$, for every instance with a perfect prediction, i.e., when $\omega = v(S^*(\mathbf{c}))$.
- *Robustness:* M_τ is β-robust if $\beta \cdot \mathbb{E}\left[v(A(\mathbf{c}))\right] \geq v(S^*(\mathbf{c}))$, for every instance with an arbitrary prediction $\omega \in (0, v(S^*(\mathbf{c}))]$.

Clearly, $\alpha, \beta \geq 1$. Also, note that the expectation in each of the two benchmarks above accounts for both the randomness of M_τ and the randomness of the RO model. In addition to consistency and robustness, we also assess the performance of mechanisms when the prediction is "approximately" correct. To this end, we introduce an *error parameter* ε that quantifies how far the prediction is from $v(S^*(\mathbf{c}))$. Formally, for an instance I^+, we define the constant $\varepsilon \in [0, 1)$ to be such that $\omega = (1 - \varepsilon) \cdot v(S^*(\mathbf{c}))$, or equivalently, $\varepsilon = 1 - \omega / v(S^*(\mathbf{c}))$.

3.2 Summary of our contributions

We have initiated the study of budget-feasible mechanism design with predictions, extending essentially the original standard setting of budget feasible mechanisms without predictions by [5]. We focus on the online version of the problem in the random order model, where the agents (i.e., bidders) arrive in a uniformly random order, and under a submodular valuation function for the auctioneer. Further, the input of our mechanisms is augmented by a prediction for the value of the optimal solution to the offline problem. Our main results can be summarized as follows:

- We have proposed a family of universally truthful, budget-feasible mechanisms, parameterized by an error tolerance parameter τ for monotone submodular objectives which attain $O(1)$ consistency and robustness. For τ close to 1, the consistency, i.e., the approximation guarantee when the prediction is perfect, can be as low as 6, which is a significant improvement against the best known approximation guarantee of 54.4 for Submodular Knapsack Secretary, i.e., the non-strategic version of the problem (due to [2]). We also present a different mechanism, where the prediction implicitly determines the length of the initial agent-sampling window. Compared to our first family of mechanisms, this approach is more conservative, in the sense that it is less sensitive to (very) erroneous predictions.
- We obtain analogous results for non-monotone submodular objectives, albeit with some loss on approximation. Non-monotone submodular valuation functions do pose additional challenges, as the monotonicity of a valuation function is used crucially in the mathematical analysis. Nevertheless, our approaches can be adapted for this setting as

well. We emphasize that again our analysis leads to a large improvement on the current state-of-the-art approximation.

- Finally, we complement our positive results for the online version, by showing that for the *offline* version, access to predictions is mostly ineffective in improving approximation guarantees. In particular, we show that no randomized, universally truthful, budget-feasible mechanism with bounded robustness can achieve a consistency guarantee better than the lower bound of 2 that is known to hold for additive valuations by [1].

4 Learning-augmented Algorithms for NP-hard Permutation Problems

In this section, we discuss on a different work by our research team, which is still within the context of learning-augmented algorithms. As in Section 3, an algorithm is enhanced with hints from a machine learning model (a predictor) that has been trained on past data to recognize structural patterns in the problem instances. While the field of learning-augmented algorithms has produced extensive results for online algorithms, as in Section 3, where predictions help mitigate uncertainty about future inputs, in the current section we apply this setting to offline approximation problems. In this context, we do not predict the future, but rather the components of a computationally hard solution. Since machine learning models are heuristic in nature and do not provide worst-case guarantees, their output is not always accurate. Consequently, any algorithm utilizing these predictions must be designed with care.

Furthermore, our approach diverges from standard learning-augmented methods by enforcing parsimony. We do not blindly consume all available predictions for a given problem instance; instead, we utilize the predictor selectively. This is motivated by the fact that obtaining high-quality predictions (e.g., running a large neural network) can be computationally expensive. By limiting the number of queries made to the predictor, we aim to balance the cost of information against the gain in solution quality.

Specifically, we focus on a broad class of optimization problems where feasible solutions can be represented as permutations of the input. Suppose for example that there are n input elements for an optimization problem denoted by $a_1, \ldots, a_n$. Then a permutation (ordering) σ corresponds to the solution $(a_{\sigma(1)}, \ldots, a_{\sigma(n)})$.

4.1 Model, notation and problem definitions

Prediction model. In this work, we adopt the following probabilistic framework. For any pair of distinct elements a_i and a_j, the algorithm can get a prediction query $q(a_i, a_j)$ that indicates whether a_i precedes a_j in a fixed optimal permutation. We assume that each prediction is independently correct with probability at least $1/2+\epsilon$, for some constant $\epsilon > 0$. In total, the algorithm has potential access to $\binom{n}{2}$ such predictions, covering all possible pairs of input elements.

Our framework relies on the availability of reasonably accurate predictions for the optimal solutions of hard optimization problems. While this may seem a strong assumption, a significant body of recent research focuses on training models to predict structural properties of solutions for discrete optimization tasks. Consequently, it is reasonable to anticipate that machine learning models will increasingly be capable of acting as the predictors required by our model, providing useful heuristic guidance even if they cannot solve the problem exactly.

Furthermore, there are specific restricted settings where the derivation of such noisy predictions is theoretically well-founded. For instance, consider a scenario where problem instances are stationary, and the optimal permutation for a given instance behaves as a noisy sample from a Mallows distribution. In this context, the distribution is centered around a "mean" optimal permutation π^*, with a parameter β governing the variance. By learning this distribution from historical data, one can generate noisy rankings that serve as predictions. These learned rankings provide a concrete example of how one can obtain the necessary probabilistic guarantees to apply our approach.

Permutation problems. The designed framework is capable of handling permutation optimization problems that exhibit one of two key structural properties: the *decomposition* property and the *c-locality* property in their objective function. The decomposition property states that solving a subproblem $\mathcal{I}(i, j)$, defined between positions i and j in the permutation, optimally depends only on the set of elements assigned to positions in $[i, j]$, the permutation $\sigma(i, j)$ of these elements, and the specific sets of elements to the left of i and to the right of j, but notably *not* on the specific ordering of those external elements. On the other hand, the c-locality property states that the cost function of the problem depends only locally (with respect to the permutation) on pairs of distinct elements; specifically, the cost contribution of an element is determined solely by its neighbors within a distance c. To illustrate these properties, we examine several well-known NP-hard problems that cannot be solved exactly in polynomial time without predictions unless $P = NP$.

We first consider problems satisfying the decomposition property, beginning with graph-based ordering problems. In the **Maximum Acyclic Subgraph (MAS)** problem, we are given a simple directed graph $G = (V, E)$. The goal is to find a permutation $\sigma : V \to \{1, \ldots, n\}$ which maximizes the number of edges consistent with the ordering, defined as $f(\sigma) = |\{(u, v) \in E : \sigma(u) < \sigma(v)\}|$. Similarly, we address the **Minimum Linear Arrangement (MLA)** problem. Let $G = (V, E)$ be a simple undirected graph where the weight of an edge is defined as the absolute difference between the positions assigned to its endpoints in σ. MLA consists of finding a permutation of the vertices such that the sum of these edge weights is minimized. Formally, we seek a permutation σ that minimizes $\sum_{\{u,v\}\in E} |\sigma(u) - \sigma(v)|$.

Another representative of the decomposition property is the **Scheduling** problem, denoted as $1|prec|\sum C_j$ in standard notation. This problem is known to be strongly NP-hard. We are given a set J of n jobs and a set of precedence constraints forming a Directed Acyclic Graph (DAG) (J, A). The objective is to determine an optimal processing order for the jobs that respects all precedence constraints such that the sum of the completion times of all jobs, $\sum C_j$, is minimized.

Turning to the class of c-local problems, the most prominent example is the **Traveling Salesperson Problem (TSP)**. Given a complete graph on a set V of vertices and a distance function $d(u, v)$ between pairs of vertices, TSP asks to find a permutation σ that minimizes the total tour length $\sum_{i=1}^{n} d(v_{\sigma(i)}, v_{\sigma(i+1)})$ (where $v_{\sigma(n+1)}$ is $v_{\sigma(1)}$). Since the cost contribution of any vertex $v_{\sigma(i)}$ depends strictly on its immediate neighbor $v_{\sigma(i+1)}$, this problem is trivially 1-local.

Finally, our framework is also applicable to the problem of **Social Welfare Maximization in Keyword Auctions with Externalities**. We consider a set $N = \{1, \ldots, n\}$ of advertisers competing for a set of n advertisement slots. Each advertiser i has a valuation v_i per click and an intrinsic click probability q_i, while each slot j has a click-through rate (CTR) λ_j. In the standard model, the goal is to find an assignment π maximizing the expected social welfare $\sum v_i \lambda_{\pi(i)} q_i$. However, in the more complex *Exter-*

nalities model, the CTR of an ad i at slot $\pi(i)$ is influenced by the ads appearing in the c slots immediately above it, modeling a user's window of attention. Letting $Q_i(\pi)$ denote the conditional CTR of ad i given these predecessors, the objective becomes maximizing $\sum_{i=1}^{n} v_i \lambda_{\pi(i)} Q_i(\pi)$. This dependency on a fixed window size makes the problem naturally c-local.

4.2 Summary of our contributions

We introduced a novel learning-augmented framework designed to solve NP-hard permutation optimization problems exactly in polynomial time by leveraging noisy, pairwise predictions about the optimal ordering of elements. Our contributions are summarized as follows:

- We identified two key structural properties, the *decomposition* property and *c-locality*, and proved that any permutation problem exhibiting either of these properties can be solved exactly with high probability in polynomial time. Crucially, our approach is *parsimonious*: it does not require querying all pairs but relies on only $O(n \log n)$ prediction queries (where each prediction is correct with probability $1/2 + \epsilon$) to construct the solution.
- We demonstrated the versatility of our framework by applying it to a broad range of well-known NP-hard problems. We showed that the **Maximum Acyclic Subgraph**, **Minimum Linear Arrangement**, and **Scheduling** $(1|prec| \sum C_j)$ problems satisfy the decomposition property, while the **Traveling Salesperson Problem (TSP)** and **Social Welfare Maximization in Keyword Auctions with Externalities** satisfy the c-locality property.
- The proof of the main theorem in our work demonstrates that, for the permutation problems under consideration, knowing an approximation of each $\sigma^*(i)$ (in an optimal solution σ^*) within an additive $O(\log n)$ bound is sufficient to solve the problem in polynomial time. First, we showed that this $O(\log n)$ approximation is not always sufficient for polynomial time solvability of general permutation problems. Finally, we proved that the $O(\log n)$ bound is tight, as there are decomposable and c-local problems where an additive approximation of $f(n) \log n$ is not enough to solve these problems in polynomial time, for any unbounded function f.

5 Mechanisms for Online Fair Division of Indivisible Resources

Fair Division concerns the family of problems, where a central entity needs to allocate in a fair manner a set of resources to a set of interested agents. Since the formal introduction of these questions, numerous variants, each under different assumptions and constraints, have been studied in mathematics, economics, political science and computer science. Specifically, the algorithmic nature of most questions one may ask about fair division problems has lead to a flourishing literature on the topic, often on variants that deal with discrete, indivisible resources.

In the most standard variants of the problem a set of resources and a set of agents, each equipped with a valuation function, are given as input and one would like to produce a partition of all the resources to the agents, so that some *fairness criterion* is satisfied.

Some of the popular fairness criteria include envy-freeness up to one item (EF1) or envy-freeness up to one item (EFX), as defined in Section 5.1. Here we consider a version of the problem where the set of n agents is indeed given, but the indivisible goods arrive in an *online* fashion. That is, at each time step a new good appears, its value for the agents is revealed, and the good must be irrevocably assigned to a single agent before the next good arrives. Online fair division problems of similar nature appear in many real-world scenarios, like, e.g., the operation of food banks, donation distribution in disaster relief, limited resource allocation within an organization like a hospital or a university, or memory and CPU management in cloud computing. Despite their wide applicability, however, online fair division settings have not been studied nearly as much as their offline counterparts. And admittedly, there is a good reason for the relative scarcity of such works. Namely, it is relatively easy to show strong impossibility worst-case results, even if one aims for rather modest fairness guarantees.

Given the above, we explore a relaxation of the problem that aims to alleviate the lack of information due to the online arrival of the goods. Contrary to what we considered in Section 3 and Section 4, with the addition of possibly erroneous predictions, we consider a different model here. Instead, we allow some of our algorithms to see into the future only for a limited number of time steps. This is referred to as *lookahead* information, meaning that the algorithm designer can observe or find out the value of the goods that will arrive in the next few future steps, before deciding how to allocate the good that arrived in the current time step. In our setting this foresight is powerful enough for obtaining algorithms with fairness guarantees.

5.1 Model, notation and relevant definitions

Let $N = [n] = \{1, 2, \ldots, n\}$ be set of agents and $M = \{g_1, g_2, \ldots, g_m\}$ be a set of indivisible goods, where $n, m \in \mathbb{N}$. The high-level goal is to assign the goods of M to the agents of N in a way that is considered fair. Formally, an allocation $(A_1, \ldots, A_n)$ is an ordered tuple of disjoint subsets of M, where the set A_i is the *bundle* allocated to agent i.

Each agent i is associated with an *additive* valuation function $v_i : 2^M \to \mathbb{R}_{\geq 0}$, where $v_i(S) = \sum_{g \in S} v_i(\{g\})$ represents the value of agent i for the subset S of goods. When S is a singleton, we usually write $v_i(g)$ rather than $v_i(\{g\})$. Of course, valuation functions of the agents can be more general but in light of the known impossibility results, we only study special cases of additive instances, i.e., instances where all agents have additive valuation functions that are restricted in some way. In particular, we mostly consider instances where each agent i can have two possible values for the goods: a low value β_i and a high value α_i. We refer to these instances as *personalized 2-value instances*. It should be noted that this type of restriction has drawn significant attention in the fair division literature recently, along with its generalization to k-value instances for small values of k.

Definition 5.1 (Personalized 2-Value Instances)**.** *We say that an instance of the problem is a* personalized 2-value instance*, if for any $i \in N$ the function v_i is additive and there are $\alpha_i \geq \beta_i \geq 0$, such that for any $g \in M$, it holds that $v_i(g) \in \{\alpha_i, \beta_i\}$.*

The next definition aims to capture the continuous analog of 2-value instances, i.e., we would like all the values agent i may have for a good to be in the interval $[\beta_i, \alpha_i]$. However, as scaling the valuation functions is irrelevant for the fairness notions we consider, any general additive instance can be trivially transformed to an equivalent instance where, for any $i \in N$ and $g \in M$, it holds that $v_i(g) \in [0, 1]$. Thus, in order for the restriction

to be meaningful in our context, it should hold that $\beta_i > 0$ for all $i \in N$. By scaling appropriately, this is equivalent to asking that $\beta_i = 1$ for all $i \in N$.

Definition 5.2 (Personalized Interval-Restricted Instances)**.** *We say that an instance of the problem is a* personalized interval-restricted instance*, if for any $i \in N$ the function v_i is additive and there is $\alpha_i > 1$, such that for any $g \in M$, it holds that $v_i(g) \in [1, \alpha_i]$. When $\alpha_i = \alpha$, for all $i \in N$, we call this an* interval-restricted instance*.*

In standard—offline—fair division settings, all the goods of M and the values of the agents are known and available as input to the algorithm designer. Here we consider an *online* fair division setting, where the goods arrive one at a time; the set M (or even its cardinality, m) is not known a priori. When a good g arrives, its value for each agent is revealed and it needs to be added to the bundle of some agent immediately and irrevocably. In general, we associate a distinct time step with each good.

Our goal has been to follow a worst-case analysis, as is most commonly the case. In doing so, we also assume in some of our results that our instances can be augmented with limited information about the future. We say that an online instance is augmented with *foresight of length ℓ* if every time a good g arrives (and still needs to be allocated immediately and irrevocably), we also get a preview of the ℓ next goods.

Fairness criteria. We formalize this by introducing our main fairness notions, *approximate EFk* and *approximate MMS*. Envy-freeness up to k goods (EFk) is a relaxation of envy-freeness. For instance, according to EF1 some envy is acceptable, as long as it can be eliminated by the hypothetical removal of a single good.

Definition 5.3 (ρ-Envy-Freeness, ρ-EFk)**.** *Given an allocation $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, constants $\rho \in (0, 1]$ and $k \in \mathbb{Z}_{>0}$, and two agents $i, j \in N$, we say that*

- *agent i* is ρ-envy-free (ρ-EF) towards *agent j, if $v_i(A_i) \geq \rho \cdot v_i(A_j)$;*
- *agent i* is ρ-EFk towards *agent j, if there is a set $S \subseteq A_j$ with $|S| \leq k$, such that $v_i(A_i) \geq \rho \cdot v_i(A_j \setminus S)$.*

The allocation is called ρ-EF (resp. ρ-EFk) if every agent $i \in N$ is ρ-envy-free (resp. ρ-EFk) towards any other agent $j \in N$. When $\rho = 1$, we drop the prefix and write EFk rather than 1*-EFk.*

A different class of fairness criteria are the so called share-based notions. Maximin share fairness is such a concept, like proportionality, and can be interpreted via a thought experiment inspired by the famous cut-and-choose protocol. The idea is to give to each agent at least as much value as she could get if she partitioned the goods into disjoint sets (as many as the agents) and kept the worst among them.

Definition 5.4 (ρ-PROP, ρ-MMS)**.** *For an allocation $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, let $\Pi(n, \mathcal{A})$ be the set of possible partitions of the set $S = \bigcup_{j=1}^{n} A_j$ into n subsets. Given $\mathcal{A}$ above, a constant $\rho > 0$, and an agent $i \in N$, we say that*

- $\mathcal{A}$ is ρ-proportional *for agent i, if $v_i(A_i) \geq \rho \cdot v_i(S)$;*
- $\mathcal{A}$ is ρ-MMS *for agent i, if $v_i(A_i) \geq \rho \cdot \boldsymbol{\mu}_i^n(S)$, where $\boldsymbol{\mu}_i^n(S) = \max_{\mathcal{B} \in \Pi(n, \mathcal{A})} \min_{B_j \in \mathcal{B}} v_i(S)$ is the* maximin share *of agent i; when n is clear and S is a function of time t, we write $\boldsymbol{\mu}_i(t)$ instead $\boldsymbol{\mu}_i^n(S)$.*

The allocation is called ρ-PROP (resp. α-MMS) if it is ρ-proportional (resp. α-MMS) for every agent $i \in N$. When $\rho = 1$, we write MMS rather than 1-MMS.

It is known, and very easy to derive from the definitions, that ρ-envy-freeness implies not only ρ-envy-freeness up to k goods, for any $k > 0$, but also ρ-proportionality, which itself implies ρ-maximin share fairness. Furthermore, since our problem is online, we do not only care about the final (complete) allocation. As a result, we are interested in establishing properties of the form "the allocation at time step t is ρ_1-EFk (resp. ρ_2-MMS)" meaning that we consider and evaluate the allocation that has been constructed up to time step t. If each one of the partial allocations produced by an algorithm satisfies the same fairness guarantee, then one talks about *temporal fairness*.

Definition 5.5 (Temporal Fairness)**.** *Consider a sequence of partial allocations $\mathcal{A}^t = (A_1^t, A_2^t, \ldots, A_n^t)$, for $t \in \mathbb{Z}_{\geq 0}$, such that $A_i^t \subseteq A_i^{t+1}$ for any $i \in N$ and any $t \geq 0$. If $\mathcal{A}^t$ is ρ_1-EFk (resp. ρ_2-MMS) for all $t \in \mathbb{Z}_{\geq 0}$, then we say that the sequence of allocations $(\mathcal{A}^t)_{t \geq 0}$ is* ρ_1-temporal-EFk *(resp.* ρ_2-temporal-MMS*).*

When referring to the allocation iteratively built by an algorithm, we may abuse the terminology and say that the algorithm computes a ρ_1-temporal-EFk (resp. ρ_2-temporal-MMS) allocation, rather than talking about a sequence of allocations.

5.2 Summary of our contributions

Our work provided an almost exhaustive exploration on what type of fairness guarantees are possible for (personalized) 2-value instances in online fair division. Specifically:

- The impossibility results that are known for general additive instances persist in our setting as well, albeit milder. We show that, for any $\varepsilon > 0$, no algorithm can guarantee $(1/2 + \varepsilon)$-EF1 at every time step, even for two agents, or $(1/(2n-1) + \varepsilon)$-MMS at every time step for general n.

- We demonstrate that even very limited look-ahead knowledge of the future may help significantly. We obtain an algorithm that guarantees EF1 at every *other* time step for two agents just by looking one step ahead into the future.

- More generally, we show that having a foresight of $n-1$ goods into the future suffices in order to guarantee EF2 at every time step and EF1 at every n time steps for n agents. The latter also implies a $1/n$-MMS approximation at every n time steps, whereas achieving $(1/n + \varepsilon)$-MMS at every time step is impossible, even if the whole instance is fully known in advance.

- Without foresight, we present an algorithm with a tight approximation guarantee with respect to MMS. In particular we can obtain a guarantee of $1/(2n-1)$-MMS at every time step, which improves to $\Omega(1)$, assuming that m is large enough.

- Finally, we provide a simple reduction that allows us to translate our results to general additive instances at the expense of a multiplicative factor that depends on the largest ratio between the values of any two goods. To the best of our knowledge, these are the first positive results in this setting.

References

[1] Ning Chen, Nick Gravin, and Pinyan Lu. On the approximability of budget feasible mechanisms. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 685–699. SIAM, 2011.

[2] Moran Feldman, Joseph Naor, and Roy Schwartz. Improved competitive ratios for submodular secretary problems. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 218–229. Springer, 2011.

[3] Alexander Lindermayr, Nicole Megow, Bertrand Simon, Adam Polak, and Niklas Hahn. Algorithms with predictions site (ALPS). `https://algorithms-with-predictions.github.io/`.

[4] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4), July 2021.

[5] Yaron Singer. Budget feasible mechanisms. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010,*, pages 765–774, 2010.

A Appendix

In the remainder of this deliverable we include the three articles that are discussed in Sections 3, 4 and 5.

Online Budget-Feasible Mechanism Design with Predictions

Georgios Amanatidis[1,3], Evangelos Markakis[1,3,5], Christodoulos Santorinaios[1,3], Guido Schäfer[2,4], Panagiotis Tsamopoulos[1], and Artem Tsikiridis[2]

[1] Athens University of Economics and Business, Greece
[2] Centrum Wiskunde & Informatica (CWI), The Netherlands
[3] Archimedes, Athena Research Center, Greece
[4] University of Amsterdam, The Netherlands
[5] Input Output Global (IOG), Greece

Abstract. Augmenting the input of algorithms with predictions is an algorithm design paradigm that suggests leveraging a (possibly erroneous) prediction to improve worst-case performance guarantees when the prediction is perfect (consistency), while also providing a performance guarantee when the prediction fails (robustness). Recently, Xu at al. [40] and Agrawal et al. [1] proposed to consider settings with strategic agents under this framework. In this paper, we initiate the study of budget-feasible mechanism design with predictions. These mechanisms model a procurement auction scenario in which an auctioneer (buyer) with a strict budget constraint seeks to purchase goods or services from a set of strategic agents, so as to maximize her own valuation function. We focus on the *online* version of the problem where the arrival order of agents is random. We design mechanisms that are truthful, budget-feasible, and achieve a significantly improved competitive ratio for both monotone and non-monotone submodular valuation functions compared to their state-of-the-art counterparts without predictions. Our results assume access to a prediction for the value of the optimal solution to the offline problem. We complement our positive results by showing that for the offline version of the problem, access to predictions is mostly ineffective in improving approximation guarantees.

Keywords: procurement auctions · online mechanisms · budget feasibility · prediction-augmented mechanisms.

1 Introduction

Our work evolves around the design of auction mechanisms for procurement auctions. Procurement auctions, also known as reverse auctions, are mechanisms for assigning a set of projects or tasks to a set of candidate providers. In the more traditional applications of reverse auctions, the auctioneer may be a governmental organization (e.g., for assigning a public project to a construction firm), or a company interested in outsourcing tasks, such as the supply of logistics or retail

services; see [21]. In recent years, the explosion and monetization of the Internet, have created even further applications and platforms where even a single individual can conduct online assignments/auctions. Two such prominent families of examples include platforms for online labour markets, such as Upwork or Freelancer, and platforms for crowdsourcing systems, where the posted tasks can range from very elementary assignments as image labeling, all the way to more specialized services like translating documents. Participatory crowdsensing also falls under this framework, where some entities are interested in collecting sensing information from smartphones or other devices.

The emergence of such online markets, has caught the increasing attention of the research community, which has focused on properly formulating and addressing relevant research questions. One of the popular models in the literature for capturing procurement auctions is that of budget-feasible mechanism design, originally proposed in the seminal paper of Singer [38]. This model concerns a scenario where an auctioneer with a strict budget constraint seeks to purchase goods or services from a set of strategic agents, who may misreport their cost to their advantage for obtaining higher payments. Under these considerations, a natural goal for the auctioneer is to come up with a truthful mechanism for hiring a subset of the agents that maximizes the procured value (measured by the valuation function of the auctioneer), and such that the total payments to the agents respect the budget limitation. Given that even the non-strategic versions of such budget-constrained problems tend to be NP-hard, the main focus is on providing budget-feasible mechanisms that achieve approximation guarantees on the auctioneer's optimal potential value. This has led to a steady stream of works which we outline in our related work section.

The focus of our work is to study budget-feasible mechanisms under the rather recently introduced framework of learning-augmented mechanism design. This forms a new paradigm in designing algorithms, where the input is augmented by a prediction on some relevant parameter of the problem, possibly obtained by past data. The main goal is to leverage a (possibly erroneous) prediction to improve worst-case performance guarantees when the prediction is accurate (referred to as consistency), while also providing a performance guarantee when it is not (referred to as robustness), without knowing the prediction accuracy in advance. Recently, settings with strategic agents were considered through this lens [40,1]. Motivated by these two initial attempts, there has been a fast growing interest and a series of other works, that studied the design of truthful mechanisms with predictions in a variety of settings, leading mostly to positive results (i.e., improved approximation guarantees under correct predictions while not far off from the best known guarantee otherwise).

Our Contribution. We initiate the study of budget-feasible mechanism design with predictions. We focus on the online version of the problem in the random order model, where the agents arrive in a uniformly random order, and under a submodular valuation function for the auctioneer. Further, the input of our mechanisms is augmented by a prediction for the value of the optimal solution to the offline problem. Our main results can be summarized as follows:

- In Section 3 we propose a family of universally truthful, budget-feasible mechanisms (Mechanism 1; parameterized by an error tolerance parameter τ) for monotone submodular objectives which attain $O(1)$ consistency and robustness. For τ close to 1, the consistency, i.e., the approximation guarantee when the prediction is perfect, can be as low as 6, which is a significant improvement even versus the best known approximation guarantee of 54.4 for Submodular Knapsack Secretary, i.e., the non-strategic version of the problem, without predictions (due to [22]).
- For smaller values of τ, the robustness of Mechanism 1 in Section 3, i.e., the approximation guarantee when the prediction is arbitrarily bad, can be as low as 146. This also implies a 146-approximation mechanism *without predictions*, improving the previously known approximation of 1710 of [2] by a factor greater than 11!
- In Section 4, we present a different mechanism (Mechanism 4) where the prediction implicitly determines the length of the initial agent-sampling window (although here we have fixed the latter for the sake of presentation). Compared to Mechanism 1, this approach is more conservative, as Mechanism 4 is, in a sense, less sensitive to (very) erroneous predictions. In particular, we show that our mechanism achieves a consistency-robustness tradeoff of 95 and 280.
- We obtain analogous results for non-monotone submodular objectives, albeit with some loss on approximation. Specifically, for τ close to 1, the consistency of our mechanism can be as low as 19, whereas, for smaller values of τ, its robustness can be as low as 445. We emphasize once more that, even when one completely ignores the prediction, our analysis leads to a large improvement on the current state-of-the-art approximation [2], which is 1710 even for Submodular Knapsack Secretary (the non-strategic version of the problem). Finally, the analogue of Mechanism 4 for non-monotone submodular valuations leads to a consistency-robustness tradeoff of 228 and 818.
- We complement our positive results by showing in Section 6 that for the *offline* version, access to predictions is mostly ineffective in improving approximation guarantees. In particular, we show that no randomized, universally truthful, budget-feasible mechanism with bounded robustness can achieve a consistency guarantee better than the lower bound of 2 that is known to hold for additive valuations and randomized mechanisms [17].

All our mechanisms run in polynomial time assuming access to a value oracle for the objective function, as is standard in submodular optimization. From a technical point of view, we design posted price auction mechanisms, the prices of which are determined by exploiting the information from the prediction on the optimal value and the outcome of an initial sampling phase. Overall, our results shed light on the power but also on the limitations of the learning-augmented paradigm for budget-feasible mechanisms.

All missing proofs from this extended abstract are deferred to the full version of our paper [3].

Related Work. The design of truthful budget-feasible mechanisms was initiated by Singer [38] for additive and monotone submodular valuation functions, and

has sparked a rich line of work. For additive valuation functions a best-possible 2-approximation randomized mechanism was given by Gravin et al. [26], whereas it is also known that the best possible factor of any deterministic mechanism is between $1+\sqrt{2}$ [17] and 3 [26]. For monotone submodular valuation functions, through a series of improvements (see e.g., [17,29,7]), the current state of the art is 4.45 for deterministic and 4.08 for randomized mechanisms due to Han et al. [27] who follow the clock auction paradigm. Without the monotonicity assumption, obtaining any $O(1)$ approximation seemed to be more challenging (see e.g., [2]), but here as well recent improvements [7,28] have led to a relatively small approximation ratio, namely 11.67 [27]. We refer the reader to the recent survey of Liu et al. [32] for an overview of further variants of the problem.

A recent direction in theoretical computer science is the area of "beyond the worst-case analysis" (see, e.g., [37]). One prominent approach within this framework is *algorithms with predictions*, which originated in the field of online algorithms (see, e.g., [34]) and utilizes predictions, e.g., provided by an ML algorithm trained on historical data, to overcome worst case lower bounds. The idea has been also applied in strategic settings, where predictions are used to simulate (an aggregation of) the agents' private information, starting with the work of [1] and [40]. Areas of application in mechanism design include auction-related environments [36,10,18,15,9,25], online mechanisms [10,33], mechanisms without money [8,20,12,19] and computational social choice [23]. For an almost exhaustive list of papers in the area, see the online repository of [31].

The *online* budget-feasible mechanism design problem we study was introduced by Badanidiyuru et al. [6] and is a generalization to the strategic setting of the Submodular Knapsack Secretary problem (see, e.g., [5]). The latter has also been considered in [13] and subsequently in [22,30]. Very recently, [16] devised a truthful mechanism which achieves a constant approximation (for a very large constant) for *posted pricing* and monotone submodular valuations. The state-of-the-art approximation for monotone or non-monotone submodular valuation functions for the strategic version we study here is due to [2], who achieve a competitive ratio of 1710. Note that the Secretary problem (additive objective and cardinality constraint) has been studied in environments with predictions (see, e.g., [4,14,24,11]), yet our work is the first to study the online budget-feasible mechanism design problem—and its algorithmic counterpart—in such augmented environments.

2 Preliminaries

We explore online mechanism design with predictions, focusing on a procurement auction setting involving a single auctioneer and multiple agents. The auctioneer has a valuation function $v : 2^N \to \mathbb{Q}_{\geq 0}$ and a budget $B > 0$. We use $N = [n] = \{1, 2, \ldots, n\}$ to denote the set of n agents. Each agent i has a *private* cost $c_i > 0$, namely the cost of getting hired by the auctioneer. For $S \subseteq N$, $v(S)$ represents the value the auctioneer derives from selecting that set; for singletons, we write $v(i)$ instead of $v(\{i\})$. The algorithmic goal in all

the problems we study is to select a set S that maximizes $v(S)$ subject to the constraint $\sum_{i\in S} c_i \leq B$. We assume oracle access to v via value queries, implying the existence of a polynomial-time value oracle that returns $v(S)$ when queried with a set S. A function v is non-decreasing (referred to as *monotone* herein), if $v(S) \leq v(T)$ for any $S \subseteq T \subseteq N$. We consider the cases of monotone and non-monotone, normalised (i.e., $v(\emptyset) = 0$), non-negative *submodular* valuation functions. Since marginal values are extensively used, we adopt the shortcut $v(i|S)$ for the marginal value of agent i with respect to the set S, i.e., $v(i|S) := v(S \cup \{i\}) - v(S)$.

Definition 1. *A function v, defined on 2^N for some set N, is submodular if $v(i|S) \geq v(i|T)$ for all $S \subseteq T$ and $i \notin T$.*

Note that when $v(i|S) = v(i|\emptyset)$, for all $i \in N$ and all $S \subseteq N$, the valuation function v is additive. The following will be useful in Section 3.

Theorem 1 ([35]). *A function v is submodular if and only if, for all $S, T \subseteq N$*

$$v(T) \leq v(S) + \textstyle\sum_{i\in T\setminus S} v(i|S) + \sum_{i\in S\setminus T} v(i|(S\cup T)\setminus\{i\}).$$

An important element in our problem is the online arrival of agents. We adopt the *random order* (RO) model; see also the discussion in the beginning of Section 3. In this model an adversary first chooses the costs of all the agents in N. There are n different time slots $t_1, \ldots, t_n$, with a single agent appearing in each slot according to a uniformly random permutation π, i.e., the input sequence to an *online* algorithm is $\pi(1), \pi(2), \ldots, \pi(n)$. Such an algorithm has to perform its irrevocable actions for agent $\pi(i)$ before seeing agent $\pi(i+1)$. The length n of the input sequence may be known in advance, and we assume so in our setting. For the RO model, given the output A of an algorithm and the value of an optimal solution OPT, we define the competitive-ratio (for maximization problems) to be $\frac{OPT}{\mathbb{E}[v(A)]}$ on the adversarily-chosen (worst case) set of inputs. The expected value is over all permutations of the input.

Mechanism Design. An offline mechanism $M = (A, p)$ consists of an allocation function $A : \mathbb{R}^n_{\geq 0} \to 2^{[n]}$ and a payment rule $p : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$. Given a profile of cost declarations $\mathbf{b} \in [0, B]^n$, where each b_i is the cost *reported* by agent i, the allocation function selects a set of agents $A(\mathbf{b}) \subseteq N$. The payment rule returns a profile of payments $\mathbf{p}(\mathbf{b}) = (p_1, \ldots, p_n)$, where $p_i(\mathbf{b}) \geq 0$ is the payment to agent $i \in N$. Therefore, the utility of agent $i \in N$ is $u_i(\mathbf{b}) = p_i(\mathbf{b}) - c_i$ if $i \in A(\mathbf{b})$ and $u_i(\mathbf{b}) = 0$ otherwise, i.e., that agents not in $A(\mathbf{b})$ are not paid.

An *online* mechanism, on the other hand, decides the allocation and payment of an agent at every time step without having seen the entire input. That is, given a permutation π on N, the mechanism decides about agent $\pi(t)$ at time step $t \in [n]$, as $\pi(t)$ appears and declares her cost, i.e., at the time the mechanism has only seen $\mathbf{b}^\pi_t = (b_{\pi(1)}, \ldots, b_{\pi(t)}) \in [0, B]^t$. The mechanism, thus, maintains the pair $(A_t(\mathbf{b}^\pi_t), p_t(\mathbf{b}^\pi_t))$ of currently selected agents and their payments, which is consistent with the decisions being irrevocable.

For a *deterministic* mechanism $M = (A, p)$ we require that, for every arrival order, any true cost profile $\mathbf{c}$ and any declared cost profile $\mathbf{b}$, the mechanism is:

– *budget-feasible*; the payments to the agents do not exceed the auctioneer's budget, i.e., $\sum_{i=1}^{n} p_i(\mathbf{b}) \leq B$.
– *individually rational*; no agent can be hired for less than they asked for, i.e., $p_i(\mathbf{b}) \geq b_i$, if $i \in A(\mathbf{b})$.
– *truthful*; no agent $i \in N$ has an incentive to misreport their true cost c_i, i.e., $u_i(c_i, \mathbf{b}_{-i}) \geq u_i(\mathbf{b})$.

A *randomized* mechanism $M = (A, p)$ can be thought of as a probability distribution over deterministic mechanisms for every fixed arrival order. In this work, we require all our randomized mechanisms to be probability distributions over budget-feasible, individually rational, and truthful deterministic mechanisms. Randomized mechanisms with the latter property are called *universally truthful*.[6] Since all our mechanisms are provably universally truthful, we will consistently refer to the true cost profile $\mathbf{c}$ instead of the declared cost profile $\mathbf{b}$ when needed. Furthermore, we will omit referring explicitly to $\mathbf{c}$ if it is clear from the context.

Given the above, an instance of the problem is described by the tuple $I = (N, \mathbf{c}, v, B)$. A common practice in this line of work is to measure the performance of a mechanism $M = (A, p)$, both in online variants and the offline problem, against the optimal value of the following packing problem:

$$\max\ v(S) \quad \text{s.t.} \quad \textstyle\sum_{i \in S} c_i \leq B, \quad S \subseteq N. \tag{1}$$

We use $S^*(\mathbf{c}) \subseteq N$ to refer to an optimal solution of (1).

Predictions. We employ a beyond-worst-case analysis perspective to measure the performance of mechanisms. We assume that each instance $I = (N, \mathbf{c}, v, B)$ is *augmented* by a (possibly erroneous) deterministic prediction ω of $v(S^*(\mathbf{c}))$, and that a mechanism is equipped with an error *tolerance* parameter $\tau \in [0, 1)$, set by the mechanism designer, representing how much tolerance she has for the prediction's inaccuracy. We denote the augmented instance as $I^+ = (N, \mathbf{c}, v, B, \omega)$. As is standard in the algorithms with predictions literature, we assess the performance of a mechanism with tolerance τ, $M_\tau = (A, p)$, on augmented instances based on the following two metrics:

– *Consistency:* M_τ is α-consistent if $\alpha \cdot \mathbb{E}\left[v(A(\mathbf{c}))\right] \geq v(S^*(\mathbf{c}))$, for every instance with a perfect prediction, i.e., when $\omega = v(S^*(\mathbf{c}))$.
– *Robustness:* M_τ is β-robust if $\beta \cdot \mathbb{E}\left[v(A(\mathbf{c}))\right] \geq v(S^*(\mathbf{c}))$, for every instance with an arbitrary prediction $\omega \in (0, v(S^*(\mathbf{c}))]$.

Clearly, $\alpha, \beta \geq 1$. Also, note that the expectation in each of the two benchmarks above accounts for both the randomness of M_τ and the randomness of the RO model. In addition to consistency and robustness, we also assess the performance of mechanisms when the prediction is "approximately" correct. To this end, we introduce an *error parameter* ε that quantifies how far the prediction is from $v(S^*(\mathbf{c}))$. Formally, for an instance I^+, we define the constant $\varepsilon \in [0, 1)$ to be such that $\omega = (1 - \varepsilon) \cdot v(S^*(\mathbf{c}))$, or equivalently, $\varepsilon = 1 - \omega / v(S^*(\mathbf{c}))$.

[6] This is stronger than *truthfulness in expectation*, which requires that truth-telling maximizes each agent's expected utility.

Note that, throughout this work, we assume that ω is an underestimator of $v(S^*)$ and, thus, the analyses presented refer to a parameterization of the prediction as $\omega = (1-\varepsilon)\cdot v(S^*)$ with $\varepsilon \in [0,1)$. This is mainly for the sake of presentation; our results extend for $\omega = (1+\varepsilon)\cdot v(S^*)$ with $\varepsilon \in [0,\kappa]$, for constant κ, through very similar analyses.

3 A Natural Prediction-Augmented Mechanism

In this section, we assume that the valuation function v is monotone submodular. We address the more general, non-monotone case in Section 5.
We start with a short discussion about the online model we consider. Clearly, the most well-studied model for online algorithms, is the adversarial one. For our problem, that would mean that an adversary chooses the cost and value of each agent, as well as the time of their arrival. Versus such an adversary though, no meaningful approximation guarantee for consistency and robustness simultaneously is possible.

Proposition 1. *No online mechanism, augmented with a prediction for the value of an optimal solution, can provide both a bounded consistency and a bounded robustness guarantee in the adversarial model of arrival.*

Given Proposition 1, we turn our attention on the RO model of arrival, defined in Section 2. The main result of this section is Mechanism 1, which is a universally truthful, budget-feasible randomized mechanism with constant consistency and robustness guarantees that runs in polynomial time. Moreover, the tradeoff between consistency and robustness that the mechanism achieves is tunable via an error tolerance parameter $\tau \in [0,1)$. Mechanism 1 can be viewed as a convex combination between two other mechanisms, namely Mechanism 2 and Mechanism 3. The first one (Mechanism 2), is a posted price mechanism that builds a budget feasible solution based solely on the information provided by the prediction. The second one (Mechanism 3), is also a posted price mechanism which, however, ignores the prediction. This mechanism bears similarities to the mechanism in [6], however, our analysis is significantly tighter. The resulting mechanism (Mechanism 1) randomizes between the two outcomes using a tolerance parameter $\tau \in [0,1)$.

Mechanism 1: Online mechanism for a monotone submodular v, parameterized by tolerance $\tau \in [0,1)$.

1 **With probability** τ**:** Run Mechanism 2
2 **With probability** $1-\tau$**:** Run Mechanism 3

Before analyzing the approximation performance, we first establish all the other desirable properties of Mechanism 1 in the following lemma.

Mechanism 2: Online mechanism for a monotone submodular function v, augmented with a prediction ω of $v(S^*)$.

With probability p**:**
  Return the first agent j for whom $v(j) \geq \frac{a \cdot \omega}{z}$ and pay her B.
With probability $1-p$**:**
  Set $t = a \cdot \omega$
  Set $S = \emptyset$, $B' = B$
  for *each round as an agent* i *arrives* **do**
    if $c_i \leq \bar{p}_i := \frac{B}{t} v(i \mid S)$ *and* $B' - \bar{p}_i \geq 0$ **then**
      Add i to S, set $p_i = \bar{p}_i$ and $B' = B' - p_i$.
return winning set S and payments $\mathbf{p}$.

Mechanism 3: Online mechanism for a monotone submodular function v.

With probability q**:** `/* Dynkin's Algorithm */`
  Sample the first $\lfloor n/e \rfloor$ agents; let i^* be the most valuable among them.
  From the remaining agents, return the first agent j for whom $v(j) \geq v(i^*)$ and pay her B.
With probability $1-q$**:**
  Draw ξ_1 from the binomial distribution $\mathcal{B}(n, \frac{1}{2})$.
  Let N_1 be the set of the first ξ_1 agents that arrive.
  Let T_1 be a $(1 - \frac{1}{e})$-approximate solution to (1) on N_1, using the algorithm of Sviridenko [39].
  Set $N_2 = N \setminus N_1, B' = B, S = \emptyset$ and $\mathbf{p} = \mathbf{0}$.
  Set $t = \beta \cdot v(T_1)$
  for *each round as agent* $i \in N$ *arrives* **do**
    if $c_i \leq \bar{p}_i := \frac{B}{t} v(i \mid S)$ *and* $B' - \bar{p}_i \geq 0$ **then**
      Add i to S, set $p_i = \bar{p}_i$ and $B' = B' - p_i$.
return winning set S and payments $\mathbf{p}$.

Lemma 1. *Mechanism 1 is universally truthful, budget-feasible and individually rational.*

Proof. Regarding universal truthfulness, since we have a probability distribution over two different mechanisms, it suffices to argue about the truthfulness of each of them separately.

Mechanism 2, with probability p, only hires the first agent j such that $v(j) \geq \frac{a \cdot \omega}{z}$ and pays her B, which is truthful since the private information of the agents is not used and for each agent i, holds that $c_i \leq B$.

Coming now to the second part of Mechanism 2 that is run with probability $1-p$, fix an arrival sequence, as realized by the random arrival model. Note that the agents have no control over their arrival position. Any agent $i \in N_1$ is always

rejected by the mechanism, regardless of her cost. Moreover, each agent $i \in N$ is offered a price p_i, which is independent of her declared cost, since agent i has no control over her slot of arrival and thus cannot affect her marginal contribution.

For the sake of contradiction, assume that agent $j \in N$ can achieve a better outcome by misreporting a cost $b_j \neq c_j$, while all other agents keep the same reported costs. Suppose first that agent j belongs to the winning set S, under the truthful profile. Then by misreporting, she receives the same payment as long as her declared cost is below the threshold p_j which is still the same as before, since the algorithm runs in exactly the same way up until the time that i arrives. If the declared cost is above the threshold, she is rejected by the mechanism and receives a payment of 0. Hence, when $j \in S$, she cannot guarantee a better utility by misreporting her true cost. Suppose now that agent j does not belong to the solution S, under truthful reporting. She certainly cannot benefit by reporting a higher cost than c_j, as that would still result in rejection by the mechanism. Additionally, if she reports a lower cost than c_j and she is added to the solution S, this means that the reason for rejection before was that the offered payment was less than her true cost c_j (and not because of budget exhaustion). By reporting $b_c < c_j$, given that the payment offered by the mechanism remains the same, this results in negative utility for agent j. All of the above leads to the conclusion that no agent can benefit from misreporting her true cost.

In a similar manner, one can prove that Mechanism 3 is also truthful.

For budget feasibility, note that for both mechanisms, in the case that the mechanism hires the first agent that passes threshold $\frac{a \cdot \omega}{z}$ (Mechanism 2) or when Dynkin's algorithm is run (Mechanism 3), the mechanism pays exactly B. Moreover, in both mechanisms the inequality $B' - \bar{p_i} \geq 0$ guarantees that for the solution S returned by the mechanism, $\sum_{i \in S} p_i \leq B$, which ensures the budget feasibility of the solution S, in the case that Dynkin's algorithm is not selected.

Finally, through the inequality $c_i \leq \bar{p_i}$ and the fact that $c_i \leq B$ for any $i \in [N]$, individual rationality is ensured for both mechanisms, since for any agent i added to S, it holds that $p_i \geq c_i$. □

The main theorem of this section is Theorem 2 below, which provides the consistency and robustness guarantees, in terms of the involved parameters. Parameters $a \in (0,1)$, $\beta \in (0,1)$, $p \in [0,1]$ and $q \in [0,1]$ are parameters of the mechanisms, $z \in [1, \infty)$ and $\delta \in (0,1)$ are auxiliary parameters that appear only in the analysis of the mechanisms. All parameters will be set to certain values. This also applies to the rest of the paper.

Theorem 2. *Mechanism 1 with access to a prediction $\omega = (1-\varepsilon) \cdot v(S^*)$ with $\varepsilon \in [0,1)$, achieves in expectation, approximation guarantee of $(\tau \cdot f_1 + (1-\tau) \cdot f_2)^{-1}$, where f_1 is the expected approximation ratio of Mechanism 2 and is equal to $\min\left\{\frac{a \cdot (1-\varepsilon) \cdot p}{z}, (1-p) \cdot a \cdot (1-\varepsilon) \cdot \frac{z-1}{z}, (1-p)(1-(1-\varepsilon) \cdot a)\right\}$ and f_2 is the expected approximation ratio of Mechanism 3, which is $\min\{\frac{q}{e} \cdot \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta), (1-q) \cdot \tilde{p} \cdot ((\frac{1}{2} - \delta) - \beta), (1-q) \cdot \tilde{p} \cdot (\frac{z-1}{z} \cdot \beta \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta))\}$, with $\tilde{p} = 1 - 2 \cdot \exp\left(-\frac{\delta^2/2}{(\frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)) \cdot (\frac{1}{4} + \frac{\delta}{3})}\right)$.*

From Theorem 2, we obtain the following two corollaries, which gives us the in expectation consistency and robustness guarantees of Mechanism 1.

Corollary 1. *Mechanism 1, given access to a perfect prediction $\omega = v(S^*)$ achieves in expectation, consistency factor of $(\tau \cdot \frac{1}{6} + (1-\tau) \cdot \frac{1}{146})^{-1}$, by setting $p = 0.46$, $a = 0.685$, $z = 1.85$ for Mechanism 2 and $q = 0.66$, $z = 2.1$, $b = 0.29$, $\delta = 0.174$ for Mechanism 3.*

Corollary 2. *Mechanism 1, given access to an erroneous prediction $\omega = (1-\varepsilon) \cdot v(S^*)$, with $\varepsilon \in (0,1)$, achieves in expectation, robustness factor of $(\tau \cdot (\frac{1}{6} \cdot (1-\varepsilon)) + (1-\tau) \cdot \frac{1}{146})^{-1}$ by setting $p = 0.46$, $a = 0.685$, $z = 1.85$ for Mechanism 2 and $q = 0.66$, $z = 2.1$, $b = 0.29$, $\delta = 0.174$ for Mechanism 3.*

Below, we will present the proof of Theorem 2, by analysing separately the, in expectation, performance guarantees of Mechanism 2 and Mechanism 3. From now and for the rest of the paper, we will denote as $S^* = \{a_1, \ldots, a_\ell\}$ the winning set of an optimal offline solution (arbitrarily chosen if there is more than one) and also $S_1^* = N_1 \cap S^*$, $S_2^* = N_2 \cap S^*$ (for the mechanisms that sets N_1 and N_2 are used).

Lemma 2. *Mechanism 2, with access to a prediction $\omega = (1-\varepsilon) \cdot v(S^*)$ with $\varepsilon \in [0,1)$, has an in expectation approximation ratio of:*

$$\min\left\{ \tfrac{a \cdot (1-\varepsilon) \cdot p}{z}, (1-p) \cdot (1-(1-\varepsilon) \cdot a), (1-p) \cdot a \cdot (1-\varepsilon) \cdot \tfrac{z-1}{z} \right\}.$$

Proof. Let $i^* \in \arg\max_j v(\{j\})$. We will discriminate between three cases.

Case 1: Agent i^* has significant value, i.e.: $v(i^*) > \frac{a \cdot (1-\varepsilon) \cdot v(S^*)}{z}$.
For this case, we will utilize the fact that with probability p, the mechanism hires the first agent j for whom it holds that $v(j) \geq \frac{a \cdot \omega}{z} = \frac{a \cdot (1-\varepsilon) \cdot V(S^*)}{z}$. Therefore, independently of what our mechanism does with the remaining probability, the expected value for $v(S)$ will be at least

$$E[v(S)] \geq \frac{p \cdot a \cdot (1-\varepsilon)}{z} \cdot v(S^*).$$

Case 2: Each $i \in S^* \setminus S$ rejects the posted price and $v(i^*) \leq \frac{a \cdot (1-\varepsilon) \cdot v(S^*)}{z}$.
An agent i rejects the posted price $\bar{p}_i$, only if $c_i > \bar{p}_i$. Then, given that S_i is the current solution at the time when i arrives,

$$\sum_{i \in S^* \setminus S} (v(S_i \cup \{i\}) - v(S_i)) < \sum_{i \in S^* \setminus S} \frac{t}{B} \cdot c_i \leq \frac{a \cdot \omega}{B} \cdot B = a \cdot (1-\varepsilon) \cdot v(S^*),$$

where the first inequality holds, due to the fact that each agent in $S^* \setminus S$ rejected the posted price offered by the mechanism and the second from the fact that the optimal solution is feasible. This inequality combined with Theorem 1, leads us to the following:

$$v(S^*) - v(S) \leq \sum_{i \in S^* \setminus S} (v(S_i \cup \{i\}) - v(S_i)) \leq a \cdot (1-\varepsilon) \cdot v(S^*) \implies$$

$$v(S) \geq v(S^*) - a \cdot (1-\varepsilon) \cdot v(S^*) = (1 - a \cdot (1-\varepsilon)) \cdot v(S^*)$$

Since Mechanism 2 runs this part with probability $1-p$, in expectation we have a factor of at least

$$(1-p) \cdot (1 - a \cdot (1-\varepsilon))$$

Case 3: There is an agent $j \in S^* \setminus S$ such that $c_i \leq \bar{p}_i$ but also $\bar{p}_i > B'$ and $v(i^*) \leq \frac{a \cdot (1-\varepsilon) \cdot v(S^*)}{z}$.
For any agent i from $S^* \setminus S$ who accepted the posted price, it must hold that $\bar{p}_i > B'$, where B' is the remaining budget at the round in which agent i is considered. We first show that $B' \leq \frac{B}{z}$. To see this:

$$B' < \bar{p}_j = \frac{B}{t}(v(S_j \cup \{j\}) - v(S_j)) \leq \frac{B}{t} v(j) \leq \frac{B}{a \cdot (1-\varepsilon) \cdot v(S^*)} v(j) \leq \frac{B}{z},$$

where the last inequality holds due to the fact that $v(j) \leq v(i^*) \leq \frac{a \cdot (1-\varepsilon)}{z} v(S^*)$. We have also used submodularity, since $v(j|S_j) \leq v(j)$.
Since $B' \leq \frac{B}{z}$, we have that $\sum_{i \in S} \bar{p}_i \geq B - \frac{B}{z} = \frac{(z-1)B}{z}$. Then,

$$\frac{z-1}{z} \cdot B \leq \sum_{i \in S} \bar{p}_i = \sum_{i \in S} \frac{B}{t}(v(S_i \cup \{i\}) - v(S_i)) \leq B \cdot \frac{v(S)}{a \cdot \omega},$$

which leads to an expected consistency factor of

$$a \cdot (1-\varepsilon) \cdot (1-p) \cdot \frac{z-1}{z}.$$

Combining the analysis of all the three cases above, Mechanism 2, achieves in expectation a consistency of at least:

$$\min\left\{ \frac{a \cdot (1-\varepsilon) \cdot p}{ze}, a \cdot (1-\varepsilon) \cdot (1-p) \frac{z-1}{z}, (1-p)(1-(1-\varepsilon) \cdot a) \right\}.$$

This concludes the proof. □

Before continuing, we state Bernstein's concentration inequality (Theorem 3) for bounded random variables as it will be useful for the analysis that follows. We note that we use this particular concentration inequality since the sum of the random variables we define within the proof has bounded variance.

Theorem 3 (Bernstein's Inequality). *Let $X_1, \ldots, X_n$ be independent random variables such that almost surely $a_i \leq X_i \leq b_i$ and $b_i - a_i \leq C$, for any i. Consider the sum of these random variables, $S_n = \sum_{i=1}^{n} X_i$. Then, for any $\delta > 0$ it holds that*

$$Pr[|S_n - \mathbb{E}[S_n]| \geq \delta] \leq 2 \cdot \exp\left(-\frac{\delta^2/2}{V_n + C \cdot \delta/3} \right),$$

where $\mathbb{E}[S_n]$ is the expected value of S_n and V_n is the variance of S_n.

Lemma 3 will help us bound the quantity of the optimal solution that is included in sets S_1^* and S_2^* for Mechanism 3.

Lemma 3. *When* $v(i^*) \leq \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta) \cdot v(S^*)$*, for all* $i \in N$ *then with probability at least* $1 - 2 \cdot \exp\left(-\frac{\delta^2/2}{(\frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)) \cdot (\frac{1}{4} + \frac{\delta}{3})}\right)$*, it holds that* $(\frac{1}{2} - \delta) \cdot v(S^*) \leq v(S_1^*)$, $(\frac{1}{2} - \delta) \cdot v(S^*) \leq v(S_2^*)$*, where* δ *is a positive constant, that we will set to a certain value later.*

Lemma 4. *Mechanism 3, has an in expectation approximation ratio of:*

$$\min\{(1-q) \cdot \tilde{p} \cdot ((\frac{1}{2} - \delta) - \beta), (1-q) \cdot \tilde{p} \cdot (\frac{z-1}{z} \cdot \beta \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)),$$
$$\frac{q}{e} \cdot \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)\}, \text{ where } \tilde{p} = 1 - 2 \cdot \exp\left(-\frac{\delta^2/2}{(\frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)) \cdot (\frac{1}{4} + \frac{\delta}{3})}\right).$$

Proof. We discriminate between the same three cases as in the analysis of Mechanism 2.

Case 1: Agent i^* has significant value, i.e.: $v(i^*) > \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)$.
For this case, we will utilize the fact that with probability q we run Dynkin's algorithm, which gives a factor of $1/e$ over single-agent solutions. Therefore, independent of what our mechanism does with the remaining probability, the expected value for $v(S)$ is:

$$E[v(S)] \geq \frac{q}{e} \cdot \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta) \cdot v(S^*).$$

Case 2: Each $i \in S_2^* \setminus S$ rejects the posted price and $v(i^*) \leq \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)$.

$$\sum_{i \in S_2^* \setminus S} (v(S_i \cup \{i\}) - v(S_i)) < \sum_{i \in S_2^* \setminus S} \frac{t}{B} \cdot c_i = \frac{\beta \cdot v(T_1)}{B} \cdot B \leq \beta \cdot v(S^*),$$

where the first inequality holds, due to the fact that each agent in $S_2^* \setminus S$ rejected the posted price offered by the mechanism and the second from the fact that the partial solution from N_1 is upper bounded by the optimal solution. This inequality, leads us to the following one:

$$v(S_2^*) - v(S) \leq \beta \cdot v(S^*) \Leftrightarrow v(S) \geq v(S_2^*) - \beta \cdot v(S^*).$$

By plugging the bound of Lemma 3 into the above equation we get $v(S) \geq \left(\frac{1}{2} - \delta - \beta\right) \cdot v(S^*)$, with probability at least $1 - 2 \cdot \exp\left(-\frac{\delta^2/2}{(\frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)) \cdot (\frac{1}{4} + \frac{\delta}{3})}\right)$.

Case 3: There is an agent $i \in S_2^* \setminus S$ such that $c_i \leq p_i$ but $p_i > B'$ and $v(i^*) \leq \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)$.

Let $i \in S_2^* \setminus S$ be the first that has this property. We first show that $B' \leq \frac{B}{z}$. To see this:

$$B' < p_j = \frac{B}{t}(v(S_j \cup \{j\}) - v(S_j)) \leq \frac{B}{t}v(j) \leq \frac{B \cdot v(j)}{\beta \cdot v(T_1)} \leq \frac{B}{z},$$

where the last inequality holds due to the fact that $v(j) \leq v(i^*) \leq \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta) \cdot v(S^*) \leq \frac{\beta \cdot v(T_1)}{z}$ and the event of Lemma 3. We have also used submodularity, since $v(j|S_j) \leq v(j)$. Since $B' \leq \frac{B}{z}$, we have that, $\sum_{i \in S} p_i \geq B - \frac{B}{z} = \frac{(z-1) \cdot B}{z}$ and also:

$$\frac{z-1}{z} \cdot B \leq \sum_{i \in S} p_i = \sum_{i \in S} \frac{B}{t}(v(S_i \cup \{i\}) - v(S_i)) \leq B \cdot \frac{v(S)}{\beta \cdot v(T_1)},$$

which leads us to $v(S) \geq \frac{z-1}{z} \cdot \beta \cdot v(T_1) \geq \frac{z-1}{z} \cdot \beta \cdot \frac{e-1}{e} \cdot v(S_1^*)$. Which via Lemma 3 gives us that $v(S) \geq \frac{z-1}{z} \cdot \beta \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta) \cdot v(S^*)$ with probability at least $1 - 2 \cdot \exp\left(-\frac{\delta^2/2}{(\frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)) \cdot (\frac{1}{4} + \frac{\delta}{3})}\right)$.
Piecing everything together,

- For the case that $v(i^*) \leq \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot \frac{1-\delta}{2} \cdot v(S^*)$, with probability $1 - q$ we run the mechanism below line 4, which gives the factor

$$(1-q) \cdot \tilde{p} \cdot \min\left(\left((\frac{1}{2} - \delta) - \beta\right), \frac{z-1}{z} \cdot \beta \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)\right),$$

  where $\tilde{p} = 1 - 2 \cdot \exp\left(-\frac{\delta^2/2}{(\frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)) \cdot (\frac{1}{4} + \frac{\delta}{3})}\right)$.
- For the case that $v(i^*) \geq \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta)$, with probability q we run Dynkin's algorithm, which gives the factor

$$\frac{q}{e} \cdot \frac{\beta}{z} \cdot \frac{e-1}{e} \cdot (\frac{1}{2} - \delta).$$

□

By combining Lemma 2 and Lemma 4, the proof of Theorem 2 follows.

4 A Sampling-Calibrated Mechanism with Prediction Guidance

In this section, we present a second learning augmented mechanism for monotone submodular valuations, with a different approach. Rather than randomly deciding between using a prediction-based mechanism and one that ignores the prediction, we allow the designer to control the trade-off between prediction reliance and sampling through a parameter k, which determines the size of the sample set. The mechanism then combines the information obtained from sampling with the prediction to offer posted prices. When the designer places greater

trust in the quality of the prediction, a smaller sample size may suffice, as the reduced accuracy from limited sampling can be compensated for by the predicted information. This mechanism represents a more 'risk-averse' design approach; although it lacks the strong performance guarantees of Mechanism 1, its outcomes are less dependent on randomness. In particular, under a poor prediction, Mechanism 1 yields a meaningful guarantee only if the randomized choice favors Mechanism 3. In contrast, Mechanism 4 provides a non-trivial approximation guarantee even when the prediction is entirely uninformative, as it utilizes the information gained through sampling.

Mechanism 4: Online mechanism for a monotone submodular function v, parameterized by $k \geq 1$.

With probability q**:** `/* Dynkin's Algorithm */`
  Sample the first $\lfloor n/e \rfloor$ agents; let i^* be the most valuable among them.
  From the remaining agents, return the first agent j for whom $v(j) \geq v(i^*)$ and pay her B.
With probability $1-q$**:**
  Draw ξ_1 from the binomial distribution $\mathcal{B}(n, \frac{1}{k})$.
  Let N_1 be the set of the first ξ_1 agents that arrive.
  Let T_1 be $(1-\frac{1}{e})$-approximate solution to (1) on N_1, e.g., using the algorithm of [39].
  Set $N_2 = N \setminus N_1, B' = B, S = \emptyset$ and $\mathbf{p} = \mathbf{0}$.
  Set $t = a \cdot \omega + \beta \cdot v(T_1)$
  for *each round as agent* $i \in N_2$ *arrives* **do**
    if $c_i \leq \bar{p}_i := \frac{B}{t} v(i \,|\, S)$ *and* $B' - \bar{p}_i \geq 0$ **then**
      Add i to S, set $p_i = \bar{p}_i$ and $B' = B' - p_i$.
return winning set S and payments $\mathbf{p}$.

Theorem 4. *Mechanism 4, given access to a prediction* $\omega = (1-\varepsilon) \cdot v(S^*)$ *with* $\varepsilon \in [0, 1)$*, has an approximation guarantee of:*

$$\min \left\{ \frac{q}{e \cdot z} \cdot (a \cdot (1-\varepsilon) + \beta \cdot \frac{e-1}{e} \cdot (\frac{1}{k} - \delta)), (1-q) \cdot \tilde{p} \cdot \right.$$
$$\left. \cdot \min \left(\frac{k-1}{k} - \delta - (a \cdot (1-\varepsilon) + \beta), \frac{z-1}{z} \cdot (a \cdot (1-\varepsilon) + \beta \cdot \frac{e-1}{e} \cdot (\frac{1}{k} - \delta)) \right) \right\},$$

where $\tilde{p} = 1 - 2 \cdot \exp\left(- \frac{\delta^2/2}{(\frac{k-1}{k^2} + \frac{\delta}{3}) \cdot \frac{1}{z} \cdot (a \cdot (1-\varepsilon) + \beta \cdot \frac{e-1}{e} \cdot (\frac{1}{k} - \delta))} \right)$.

From the above Theorem, we derive the following corollary regarding, the in expectation consistency and robustness guarantees of Mechanism 4.

Corollary 3. *Mechanism 4, given access to a possibly erroneous prediction* $\omega = (1-\varepsilon)\cdot v(S^*)$*, with* $\varepsilon \in [0,1)$*, achieves in expectation, a consistency-robustness tradeoff of* 95 *and* 280 *by setting* $p = 0.68$*,* $a = 0.06$*,* $z = 2.15$*,* $\beta = 0.27$*,* $\delta = 0.22$ *and* $k = 2.5$*.*

5 Mechanisms for the Non-Monotone Case

Our results can be extended for general (possibly non-monotone) submodular utility functions, in the secretary model of arrival. The mechanisms follow the same approach as Mechanisms 1 and 4 for the monotone case. The difference is that for non-monotone submodular functions, it does not suffice to build a single solution in order to get any meaningful approximation guarantee. We circumvent this difficulty, by building two disjoint solutions simultaneously, which is an idea firstly introduced in [2]. Since, we are in the Random Order Model, any decision the mechanism makes upon the arrival of an agent must be irrevocable. Therefore, we are not allowed to return the best of the two solutions built and instead, the mechanism selects equiprobably, which of the two will be returned *beforehand*. These results are presented in the full version of the paper [3].

6 An Impossibility Result

Here we turn our attention to the *offline* setting and present a lower bound of 2 on the consistency guarantee for all universally truthful, budget-feasible, and individually rational randomized mechanisms *with predictions*. Clearly, this impossibility result extends to the online setting as well. Moreover, our result holds even when the instances are augmented with a prediction of the optimal set of agents $\hat{S} \subseteq N$, rather than merely the value of the optimal solution ω as assumed in Sections 3 and 5. Note that having access to $\hat{S}$ in the offline setting is at least as informative as having access to ω, since the latter can be inferred from the former, i.e., given a prediction $\hat{S}$, we may set $\omega = v(\hat{S})$. We denote by $\mathcal{M}$ the class of universally truthful, budget-feasible, and individually rational randomized mechanisms, and by $\mathcal{M}^d \subseteq \mathcal{M}$ the subclass of deterministic mechanisms that are truthful, budget-feasible, and individually rational. The main result of this section is Theorem 5.

Theorem 5. *For any constant* $\varepsilon > 0$*, no universally truthful, individually rational, budget-feasible randomized mechanism that has access to a prediction of the optimal solution can be* $(2-\varepsilon)$*-consistent, even for additive valuation functions.*

7 Discussion

In this work we initiate the study of budget-feasible mechanisms in environments with predictions for submodular valuation functions (both monotone and non-monotone). For most of the paper (Sections 3 to 5), we focus on the online

setting with random (secretary) arrivals, for which we propose two families of tunable mechanisms that incorporate, as a prediction, the value of the optimal offline solution. Our results show that mechanisms augmented with this type of relatively weak prediction can substantially boost the performance of truthful and budget-feasible mechanisms for a wide class of procurement auctions. At the same time, through our lower bound in Section 6, we obtain strong evidence that a class of natural predictions on the output are largely ineffective for the offline version of the problem with additive valuations.

For the positive results of our paper, we analyzed the environment with predictions on the value of the optimal solution. This form of prediction, also suggested by prior work (see, e.g., [10]), is relatively weak in an information-theoretic sense and can be seen as an easily obtainable aggregation of historical data (see also the discussions in [18] on output predictions). Nevertheless, we believe that studying online budget-feasible mechanism design with stronger or just different types of predictions (e.g., predictions on the set of agents that are part of the optimal solution) is an interesting direction for future work. It should be noted, however, that strong predictions that include the cost profile and would work great for the algorithmic version of the problem, here clash with the notion of truthfulness; if one designs a truthful mechanism, this part of the prediction should be purposeless.

Acknowledgments. This work has been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program. It has also been supported by the H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union – NextGenerationEU (H.F.R.I. Project Number: 15877). Finally, it was supported by the Dutch Research Council (NWO) through the Gravitation Project NETWORKS, grant no. 024.002.003 and by the European Union under the EU Horizon 2020 Research and Innovation Program, Marie Skłodowska-Curie Grant Agreement, grant no. 101034253.

References

1. Agrawal, P., Balkanski, E., Gkatzelis, V., Ou, T., Tan, X.: Learning-augmented mechanism design: Leveraging predictions for facility location. Mathematics of Operations Research **49**(4), 2626–2651 (2024)
2. Amanatidis, G., Kleer, P., Schäfer, G.: Budget-feasible mechanism design for non-monotone submodular objectives: Offline and online. Math. Oper. Res. **47**(3), 2286–2309 (2022)
3. Amanatidis, G., Markakis, E., Santorinaios, C., Schäfer, G., Tsamopoulos, P., Tsikiridis, A.: Online budget-feasible mechanism design with predictions (2025), https://arxiv.org/abs/2505.24624
4. Antoniadis, A., Gouleakis, T., Kleer, P., Kolev, P.: Secretary and online matching problems with machine learned advice. Discrete Optimization **48**, 100778 (2023)

5. Babaioff, M., Immorlica, N., Kempe, D., Kleinberg, R.: A knapsack secretary problem with applications. In: International Workshop on Approximation Algorithms for Combinatorial Optimization. pp. 16–28. Springer (2007)
6. Badanidiyuru, A., Kleinberg, R., Singer, Y.: Learning on a budget: posted price mechanisms for online procurement. In: Proceedings of the 13th ACM Conference on Electronic Commerce, EC 2012. pp. 128–145. ACM (2012)
7. Balkanski, E., Garimidi, P., Gkatzelis, V., Schoepflin, D., Tan, X.: Deterministic budget-feasible clock auctions. Operations Research (2025)
8. Balkanski, E., Gkatzelis, V., Shahkarami, G.: Randomized strategic facility location with predictions. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024)
9. Balkanski, E., Gkatzelis, V., Tan, X.: Strategyproof scheduling with predictions. In: 14th Innovations in Theoretical Computer Science Conference (ITCS 2023). Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2023)
10. Balkanski, E., Gkatzelis, V., Tan, X., Zhu, C.: Online mechanism design with predictions. In: Proceedings of the 25th ACM Conference on Economics and Computation. pp. 1184–1184 (2024)
11. Balkanski, E., Ma, W., Maggiori, A.: Fair secretaries with unfair predictions. Advances in Neural Information Processing Systems **37**, 3682–3716 (2024)
12. Barak, Z., Gupta, A., Talgam-Cohen, I.: MAC advice for facility location mechanism design. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024)
13. Bateni, M., Hajiaghayi, M., Zadimoghaddam, M.: Submodular secretary problem and extensions. ACM Transactions on Algorithms (TALG) **9**(4), 1–23 (2013)
14. Benomar, Z., Perchet, V.: Advice querying under budget constraint for online algorithms. In: Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023 (2023)
15. Caragiannis, I., Kalantzis, G.: Randomized learning-augmented auctions with revenue guarantees. In: Proceedings of the 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024. pp. 2687–2694 (2024)
16. Charalampopoulos, A., Fotakis, D., Patsilinakos, P., Tolias, T.: A competitive posted-price mechanism for online budget-feasible auctions. In: Proceedings of the 26th ACM Conference on Economics and Computation. pp. 1046–1075 (2025)
17. Chen, N., Gravin, N., Lu, P.: On the approximability of budget feasible mechanisms. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011. pp. 685–699. SIAM (2011)
18. Christodoulou, G., Sgouritsa, A., Vlachos, I.: Mechanism design augmented with output advice. In: Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024 (2024)
19. Cohen, I., Eden, A., Eden, T., Vasilyan, A.: Plant-and-steal: Truthful fair allocations via predictions. Advances in Neural Information Processing Systems **37**, 110057–110096 (2024)
20. Colini-Baldeschi, R., Klumper, S., Schäfer, G., Tsikiridis, A.: To trust or not to trust: Assignment mechanisms with predictions in the private graph model. In: Proceedings of the 25th ACM Conference on Economics and Computation. pp. 1134–1154 (2024)
21. Cramton, P., Shoham, Y., Steinberg, R.: Combinatorial Auctions. The MIT Press (2006)

22. Feldman, M., Naor, J., Schwartz, R.: Improved competitive ratios for submodular secretary problems. In: International Workshop on Approximation Algorithms for Combinatorial Optimization. pp. 218–229. Springer (2011)
23. Filos-Ratsikas, A., Kalantzis, G., Voudouris, A.A.: Utilitarian distortion with predictions. In: Proceedings of the 26th ACM Conference on Economics and Computation. pp. 254–271 (2025)
24. Fujii, K., Yoshida, Y.: The secretary problem with predictions. Mathematics of Operations Research **49**(2), 1241–1262 (2024)
25. Gkatzelis, V., Schoepflin, D., Tan, X.: Clock auctions augmented with unreliable advice. In: Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025. pp. 2629–2655. SIAM (2025)
26. Gravin, N., Jin, Y., Lu, P., Zhang, C.: Optimal budget-feasible mechanisms for additive valuations. ACM Transactions on Economics and Computation (TEAC) **8**(4), 1–15 (2020)
27. Han, K., Zhang, H., Cui, S.: Efficient and effective budget-feasible mechanisms for submodular valuations. Artificial Intelligence p. 104348 (2025)
28. Huang, H., Han, K., Cui, S., Tang, J.: Randomized pricing with deferred acceptance for revenue maximization with submodular objectives. In: Proceedings of the ACM Web Conference 2023, WWW 2023. pp. 3530–3540. ACM (2023)
29. Jalaly, P., Tardos, É.: Simple and efficient budget feasible mechanisms for monotone submodular valuations. ACM Transactions on Economics and Computation (TEAC) **9**(1), 1–20 (2021)
30. Kesselheim, T., Tönnis, A.: Submodular secretary problems: Cardinality, matching, and linear constraints. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017. LIPIcs, vol. 81, pp. 16:1–16:22 (2017)
31. Lindermayr, A., Megow, N.: Algorithms with predictions. https://algorithms-with-predictions.github.io (2025), accessed: 2025-01-19
32. Liu, X., Chan, H., Li, M., Wu, W.: Budget feasible mechanisms: A survey. In: 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024. pp. 8132–8141. International Joint Conferences on Artificial Intelligence (2024)
33. Lu, P., Wan, Z., Zhang, J.: Competitive auctions with imperfect predictions. In: Proceedings of the 25th ACM Conference on Economics and Computation, EC 2024. pp. 1155–1183. ACM (2024)
34. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. Journal of the ACM (JACM) **68**(4), 1–25 (2021)
35. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions - I. Math. Program. **14**(1), 265–294 (1978)
36. Prasad, S., Balcan, M., Sandholm, T.: Bicriteria multidimensional mechanism design with side information. In: Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023 (2023)
37. Roughgarden, T.: Beyond the worst-case analysis of algorithms. Cambridge University Press (2021)
38. Singer, Y.: Budget feasible mechanisms. In: 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010. pp. 765–774 (2010)
39. Sviridenko, M.: A note on maximizing a submodular set function subject to a knapsack constraint. Operations Research Letters **32**(1), 41–43 (2004)
40. Xu, C., Lu, P.: Mechanism design with predictions. In: Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022. pp. 571–577 (2022)

Polynomial Time Learning-Augmented Algorithms for NP-hard Permutation Problems

Evripidis Bampis [1] **Bruno Escoffier** [1] **Dimitris Fotakis** [2 3] **Panagiotis Patsilinakos** [4] **Michalis Xefteris** [5]

Abstract

We consider a learning-augmented framework for NP-hard permutation problems. The algorithm has access to predictions telling, given a pair u, v of elements, whether u is before v or not in an unknown fixed optimal solution. Building on the work of Braverman and Mossel (SODA 2008), we show that for a class of optimization problems including scheduling, network design and other graph permutation problems, these predictions allow to solve them in polynomial time with high probability, provided that predictions are true with probability at least $1/2+\epsilon$, for any given constant $\epsilon > 0$. Moreover, this can be achieved with a parsimonious access to the predictions.

1. Introduction

In recent years, advancements in Machine Learning (ML) have significantly influenced progress in solving optimization problems across a wide range of fields. By leveraging historical data, ML predictors are utilized every day to tackle numerous challenges. These developments have motivated researchers in algorithms to incorporate ML predictions into algorithm design for optimization problems. This has given rise to the vastly growing field of *learning-augmented algorithms*, also known as *algorithms with predictions*. In this framework, it is assumed that predictions about a problem's input are provided by a black-box ML model. The objective is to use these predictions to develop algorithms that outperform existing ones when the predictions are sufficiently accurate.

The idea of learning-augmented algorithms was initially introduced by Mahdian, Nazerzadeh, and Saberi, who applied it to the problem of allocating online advertisement space for budget-constrained advertisers (Mahdian et al., 2007). Later, Lykouris and Vassilvitskii formalized the framework, studying the online caching problem using predictions (Lykouris & Vassilvitskii, 2021). The main emphasis in the field of algorithms with predictions has been on online optimization, as predicting the future of a partially unknown input instance is a natural approach. However, in the past few years, the field has expanded into various other areas. An almost complete list of papers in the field can be found in (Lindermayr & Megow).

More relevant to this work, algorithms with predictions have been used to address NP-hard optimization problems and overcome their computational challenges. The first learning-augmented algorithms applied to NP-hard problems were focused on clustering, as seen in (Gamlath et al., 2022; Ergun et al., 2022; Nguyen et al., 2023), and other graph optimization problems (Chen et al., 2022). Moreover, several papers have studied MAXCUT with predictions, including (Bampis et al., 2024; Cohen-Addad et al., 2024; Ghoshal et al., 2024; Dong et al., 2025). Cohen-Addad et al. investigated the approximability of MAXCUT with predictions in two models (Cohen-Addad et al., 2024). In the first model, similar to the one used in this work, they assumed predictions for each vertex (on its position in an optimal cut) that are correct with probability $1/2 + \epsilon$, for a given $\epsilon > 0$, and presented a polynomial-time $(0.878 + \tilde{\Omega}(\epsilon^4))$-approximation algorithm. In the second model, they receive a correct prediction for each vertex with probability $\epsilon > 0$ (and no information otherwise) and designed a $(0.858 + \Omega(\epsilon))$-approximation algorithm. Ghoshal et al. also studied MAXCUT and MAX2-LIN in both models (Ghoshal et al., 2024). Furthermore, (Braverman et al., 2024) studied Maximum Independent Set within the framework of learning-augmented algorithms, adopting the aforementioned first model. Finally, (Antoniadis et al., 2024) studied approximation algorithms with predictions for several NP-hard optimization problems within a prediction model different from the one used in this work.

In this paper, we design learning-augmented algorithms for NP-hard optimization problems. Our approach does not use all available predictions for the problem at hand but instead utilizes the predictor selectively. This aligns with the con-

[*]Equal contribution [1]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France [2]National Technical University of Athens, Greece [3]Archimedes, Athena Research Center, Greece [4]Université Paris-Dauphine, Université PSL, CNRS, LAMSADE, 75016, Paris, France [5]Athens University of Economics and Business, Athens, Greece. Correspondence to: Michalis Xefteris <mxefteris@hotmail.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

cept of parsimonious algorithms, introduced by (Im et al., 2022), which aim to limit the number of predictions used, assuming that obtaining additional predictions can be computationally expensive. Here, we consider problems whose feasible solutions can be represented as permutations. There are n input elements for an optimization problem denoted by $a_1, \ldots, a_n$. A permutation (ordering) σ corresponds to the solution $(a_{\sigma(1)}, \ldots, a_{\sigma(n)})$.

Regarding the prediction model, we adopt the following probabilistic framework. For each pair i, j we can get a prediction query $q(a_i, a_j)$ that denotes whether a_i precedes a_j or not in a fixed optimal solution (permutation). Each prediction is independently correct with probability at least $1/2+\epsilon$, for a given constant $\epsilon > 0$. An algorithm has access to $\binom{n}{2}$ predictions. A formal description of the model is given in Section 2.

In this work, we use these prediction queries to solve NP-hard optimization problems exactly with high probability. We design a novel framework which is capable of handling permutation optimization problems that exhibit one of two key properties: the *decomposition* property and the *c-locality* property in their objective function (see Section 3 for formal definitions).

The decomposition property states that solving a subproblem $\mathcal{I}(i, j)$ (between positions i and j in the permutation) of the optimization problem at hand optimally depends only on the set of elements in positions in $[i, j]$, the permutation $\sigma(i, j)$ of these elements in $[i, j]$, and the set of elements to the left of i and to the right of j, but not on their order. On the other hand, the c-locality property states that the cost function of the problem depends only locally (with respect to the permutation) on pairs of distinct elements.

More specifically, we adjust and extend the approach of Braverman and Mossel (Braverman & Mossel, 2008; 2009) for the problem of sorting from noisy information and give the following theorem for a family of optimization problems (see Section 4 for its proof).

Theorem 1.1. *If the objective function of a permutation optimization problem P either exhibits the decomposition property or is c-local, then P can be solved exactly with high probability in polynomial time, using $O(n \log n)$ prediction queries.*

Therefore, if a permutation optimization problem is either decomposable or c-local, it can be solved with high probability in polynomial time within our prediction-based framework. We note that the running time depends on the accuracy of the prediction queries, i.e., on ϵ. To illustrate these properties, we examine several example problems: Maximum Acyclic Subgraph, Minimum Linear Arrangement, a scheduling problem as examples of decomposable problems, the Traveling Salesperson Problem (TSP) and social welfare maximization in keyword auctions with externalities and window size as representatives of c-local problems. All these are well-known NP-hard problems and cannot be solved exactly in polynomial time without predictions unless $\mathcal{P} = \mathcal{NP}$ (deterministic), or $\mathcal{NP} \subseteq \mathcal{BPP}$ (randomized). Moreover, the framework can naturally be extended to address a variety of other NP-hard problems with similar structural properties.

Another important aspect of our framework is that it does not query all possible pairs but instead makes only $O(n \log n)$ queries, making it parsimonious with respect to the number of predictions used. We note that this is a tight bound on the number of queries, as $\Omega(n \log n)$ queries are necessary even in the case of perfect predictions already for the Noisy Sorting Without Resampling problem (see Section 2.1). Indeed, it becomes sorting by comparisons (where comparisons are queries), for which it is well known that $\Omega(n \log n)$ comparisons are needed (even for randomized algorithms).

The proof of Theorem 1.1 demonstrates that, for the permutation problems under consideration, knowing an approximation of each $\sigma^*(i)$ (in an optimal solution σ^*) within an additive $O(\log n)$ bound is sufficient to solve the problem in polynomial time. In Section 5, we first show that this $O(\log n)$ approximation is not always sufficient for polynomial time solvability. Finally, we prove that the $O(\log n)$ bound is tight, as there are decomposable and c-local problems where an additive approximation of $f(n) \log n$ is not enough to solve these problems in polynomial time, for any unbounded function f.

We conclude this section by a brief discussion on the prediction model. Similarly as several recent articles on the topic, our framework requires good predictions of optimal solutions for hard optimization problems, which is a strong assumption. Whether ML algorithms are or will soon be able to provide such good predictions or not is currently an open question, and a large number of recent works do focus on getting predictions for discrete optimization problems (see (Bengio et al., 2021; Cappart et al., 2023) for survey-like papers, and (Khalil et al., 2017) for a specific work on predictions for problems including (Euclidean) TSP). This recent but fast-growing field makes it reasonable to hope for some accurate predictors for some permutation problems in the near future. We note also that there are restricted settings where one could outline how such noisy predictions of the optimal permutation can be learned. E.g., consider a setting where instances are stationary and instance stationarity implies that the optimal solution / permutation of each instance is a noisy sample from a Mallows distribution $\mathcal{M}(\pi^*, \beta)$ with π^* corresponding to the optimal permutation of the "mean" instance and the noise parameter β accounts for the variance of the instance distribution. Such a Mallows dis-

tribution $\mathcal{M}(\pi^*, \beta)$ can be learned from instances sampled independently from the instance distribution (Busa-Fekete et al., 2019). Samples from $\mathcal{M}(\pi^*, \beta)$ can be used as predictions in the form of noisy rankings. For fixed β, this provides whp a sufficiently good approximation of the ranking (up to an additive $O(\log n)$ term on the position of each element) so that our approach applies (see (Braverman & Mossel, 2009), where this case is discussed).

2. Background and Overview

2.1. Definitions

Formally, we consider the following probabilistic prediction model, which is inspired by the noisy query model studied in (Braverman & Mossel, 2008).

Definition 2.1. Let $A = \{a_1, \ldots, a_n\}$ and σ^* be a permutation from $[1, n]$ to $[1, n]$. For each pair (a_ℓ, a_t) in $\binom{A}{2}$ the result of a prediction query for (a_ℓ, a_t), with respect to σ^*, is $q(a_\ell, a_t) \in \{-1, 1\}$ where $q(a_\ell, a_t) = -q(a_t, a_\ell)$. We assume that:

- for each $1 \leq i < j \leq n$ the probability that $q(a_{\sigma^*(i)}, a_{\sigma^*(j)}) = 1$ is at least $\frac{1}{2} + \epsilon$, $0 < \epsilon < 1/2$,
- the queries $\{q(a_\ell, a_t) : 1 \leq \ell < t \leq n\}$ are independent conditioned on σ^*.

In this definition, the query $q(a_\ell, a_t)$ asks whether a_ℓ precedes a_t in $(a_{\sigma^*(1)}, \ldots, a_{\sigma^*(n)})$ or not. The first item states that the prediction is correct with probability at least $1/2+\epsilon$.

Given the prediction queries, we are interested in finding a permutation that maximizes the number of agreements with the queries. Formally:

Definition 2.2. Given $\binom{n}{2}$ prediction queries $q(a_\ell, a_t)$, the score $s_q(\pi)$ of a permutation $\pi : [1, n] \to [1, n]$ is given by

$$s_q(\pi) = \sum_{i<j} q(a_{\pi(i)}, a_{\pi(j)}). \tag{1}$$

We say that a permutation π^* is s-optimal if π^* is a maximizer of (1) among all permutations.

The *Noisy Sorting Without Resampling (NSWR)* problem, defined in (Braverman & Mossel, 2008), is the problem of finding an s-optimal permutation π^* with respect to a (hidden) permutation σ^* assuming that q satisfies Definition 2.1 with $p = 1/2 + \epsilon, \epsilon > 0$.

As mentioned in the introduction, we consider in this work permutation problems, i.e., optimization problems whose solutions of an instance I are permutations of n elements of a set A of I (vertices or edges in a graph, jobs in a scheduling problem,...). So the goal is to maximize or minimize $f_I(\sigma)$ for $\sigma : [1, n] \to [1, n]$. Here, $a_{\sigma(i)} \in A$ is the element of A that is in position i in the permutation (i.e., the permutation is $(a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(n)})$). To deal with feasibility constraints, $f_I(\sigma) = \infty$ if σ is unfeasible (for a minimization problem, $-\infty$ for a maximization problem).

A core part of our work will focus on permutation problems for which we have an additional information, which is an approximation of $\sigma^*(i)$ for an optimal solution σ^*. We formalize this in the following definition.

Definition 2.3. Given a permutation problem P and a permutation $(a_1, a_2, \ldots, a_n)$, P is *k-position enhanced* if we know that there exists an optimal solution σ^* such that for all i, $|\sigma^*(i) - i| \leq k$.

2.2. Framework Overview

The NSWR problem has been introduced and studied in (Braverman & Mossel, 2008). They showed the following result.

Theorem 2.4. *(Braverman & Mossel, 2008) There exists a randomized algorithm that for any $\alpha > 0$ finds an optimal solution to NSWR (s-optimal) with $p = 1/2 + \epsilon, \epsilon > 0$ in time $n^{O((\alpha+1)\epsilon^{-4})}$ except with probability $n^{-\alpha}$. Moreover, the algorithm asks $O(n \log n)$ queries.*

The proof of this theorem mainly relies on two results. The first one shows that with high probability an optimal solution π^* of NSWR (s-optimal) is close to the "hidden" permutation σ^*.

Theorem 2.5. *(Braverman & Mossel, 2008) Consider the NSWR problem, with $p = 1/2 + \epsilon, \epsilon > 0$, with respect to a permutation σ^* and let π^* be any s-optimal order assuming that q satisfies Definition 2.1. Let $\alpha > 0$. Then there exists a constant $c(\alpha, \epsilon)$ such that except with probability $O(n^{-\alpha})$ it holds that*

$$\max_i |\sigma^*(i) - \pi^*(i)| \leq c \cdot \log n = O(\log n).$$

The second result is a dynamic programming (DP) algorithm showing that k-position enhanced NSWR is solvable in $O(2^{O(k)}n^2)$. Then a specific iterative procedure allows for the computation of an optimal solution of NSWR. Very roughly speaking, the use of queries allows for a k-position enhancement for NSWR with $k = O(\log n)$ (thanks to Theorem 2.5), and then the DP algorithm works in polynomial time $O(2^{O(k)}n^2) = n^{O(1)}$.

In this work, we build upon these results to tackle various problems in our prediction setting. Roughly speaking, our framework first generates a warm-start solution using the prediction queries and then utilizes this solution to solve the problem with dynamic programming. The idea of leveraging predictions to obtain a warm-start solution has been explored in a series of papers in the literature (Dinitz et al., 2021; Sakaue & Oki, 2022). The following lemma, which makes a

connection with the aforementioned results on NSWR, will allow us to get the polynomial time algorithms claimed in Theorem 1.1.

Lemma 2.6. *Suppose that a k-position enhanced version of permutation problem P is solvable in polynomial time for $k = O(\log n)$. Then P can be solved exactly in polynomial time with high probability, using $O(n \log n)$ prediction queries.*

Proof. Let σ^* be an optimal permutation for an instance of the optimization problem P. By making $O(n \log n)$ queries, according to Theorem 2.4 we can get in polynomial time with high probability an optimal solution π^* for the NSWR problem relative to σ^*.

From Theorem 2.5, we know that with high probability $|\sigma^*(i) - \pi^*(i)| = O(\log n)$ for all i. Equivalently, for all j:

$$|\sigma^* \circ \pi^{*-1}(j) - j| = O(\log n).$$

As by assumption the k-position enhanced version of P is solvable in polynomial time for $k = O(\log n)$, we can find $\sigma^* \circ \pi^{*-1}$, hence σ^*, in polynomial time. □

Motivated by Lemma 2.6, we exhibit two sufficient conditions for a permutation problem to be polynomial time solvable when being k-positioned enhanced, for $k = O(\log n)$. These conditions (decomposability and c-locality) and their illustration on classical optimization problems are given in Section 3. The proofs that under these conditions $O(\log n)$-positioned enhanced permutation problems are polynomial time solvable are in Section 4. They are based on DP algorithms, one of which being a generalization of the one of (Braverman & Mossel, 2008).

3. Decomposability and c-locality

We now give the definitions for the decomposition property and c-locality, and illustrate them with some problems expressible in these ways. As explained before, each of these properties will allow to design DP algorithms, which is the key step to derive Theorem 1.1.

3.1. Decomposition property

To design DP algorithms, we will consider subproblems. Intuitively, for $i < j$ we will consider subproblems of finding and ordering elements in positions i to j, i.e., $(a_{\sigma(i)}, \ldots, a_{\sigma(j)})$, and need a recurrence that allows expressing the subproblem between i and j as a combination of the subproblems between i and s, and between $s+1$ and j (for $i < s < j$). A difficulty is that finding and ordering elements from positions i to j typically depends on the elements before (between positions 1 and $i-1$) and after (between positions $j+1$ and n), and on their respective ordering. Roughly speaking, our decomposition property holds when finding and ordering elements from positions i to j only depends on the *set* of elements before i and after j, not on their particular ordering.

We now formalize this idea, and illustrate it on three different problems. For a permutation $\sigma : [1, n] \to [1, n]$, we denote $\sigma(i, j)$ the subpermutation of σ on $[i, j]$, $S_\sigma(i, j)$ the set $\{\sigma(i), \ldots, \sigma(j)\}$ (indices in positions between i and j), $L_\sigma(k) = S_\sigma(1, k)$ (first k indices, L stands for left) and $R_\sigma(k) = S_\sigma(k, n)$ (indices in position between k and n, R stands for right).

Definition 3.1. Let P be a permutation problem, with objective function f. We say that P fulfills the decomposition property if there exists a function g such that:

- $f_I(\sigma) = g(1, n, \emptyset, \emptyset, \sigma)$;
- For any $i < s < j$ in $\{1, \ldots, n\}$ and permutation σ:

$$\begin{aligned} &g(i, j, L_\sigma(i-1), R_\sigma(j+1), \sigma(i, j)) = \\ &g(i, s, L_\sigma(i-1), R_\sigma(s+1), \sigma(i, s)) \\ &+ g(s+1, j, L_\sigma(s), R_\sigma(j+1), \sigma(s+1, j)) \\ &+ h(L_\sigma(i-1), R_\sigma(j+1), S_\sigma(i, s), S_\sigma(s+1, j)), \end{aligned}$$

  for some function h.

Note that h does not depend on the permutation of any elements, only on sets of elements. In this definition, we express the objective function on subproblem (i, j) by a function g that depends on the subpermutation $\sigma(i, j)$ between i and j and on sets (not positions) of elements before i and after j ($L_\sigma(i-1)$ and $R_\sigma(j+1)$).

The decomposition property states that the value is the sum of the value on the subproblem (i, s) (referred to as the "left call" in the sequel), the value on the subproblem $(s+1, j)$ (referred to as the "right call"), and a quantity (function h) that depends only on the sets (not the ordered sets) of elements involved in the decomposition.

Maximum Acyclic Subgraph. In the Maximum Acyclic Subgraph problem, we are given a simple directed graph $G = (V, E)$. The goal is to find a permutation $\sigma : [1, n] \to [1, n]$ which maximizes $f(\sigma) = |\{(v_{\sigma(i)}, v_{\sigma(j)}) \in E : i < j\}|$. This is a well known NP-hard problem (Karp, 1972).

As this problem only depends on the relative order of elements (endpoints of arcs) and not on their exact position, it is easy to express it in the form of Definition 3.1. For given $i < j$ and $\sigma(i, j)$, we simply define:

$$g(i, j, L, R, \sigma(i, j)) = |\{(v_{\sigma(\ell)}, v_{\sigma(r)}) \in E : i \leq \ell < r \leq j\}|.$$

This is just the number of arcs "well ordered" by σ with both endpoints between positions i and j (g is independent

of L and R). Obviously the first item of Definition 3.1 is verified (for $i = 1$ and $j = n$ we count the total number of arcs "well ordered" by σ).

For the second item, $g(i, s, L_\sigma(i-1), R_\sigma(s+1), \sigma(i, s))$ (resp. $g(s+1, j, L_\sigma(s), R_\sigma(j+1), \sigma(s+1, j))$) counts the number of arcs well ordered by σ with both endpoints in positions between i and s (resp. between $s+1$ and j). So, to get $g(i, j, L_\sigma(i-1), R_\sigma(j+1), \sigma(i, j))$, we only have to add arcs $(v_{\sigma(\ell)}, v_{\sigma(r)})$ with $i \leq \ell \leq s$ and $s+1 \leq r \leq j$. In other words, if we denote $cut(A, B)$ the number of arcs (v_ℓ, v_r) with $\ell \in A$ and $r \in B$, we have

$$h(L_\sigma(i-1), R_\sigma(j+1), S_\sigma(i, s), S_\sigma(s+1, j) = cut(S_\sigma(i, s), S_\sigma(s+1, j)).$$

Note that this immediately generalizes to any problem whose objective function only depends on the relative order of endpoints of arcs (for instance the weighted version of Maximum Acyclic Subgraph), not on their exact positions.

Minimum Linear Arrangement. Let $G = (V, E)$ be a simple undirected graph. Given a permutation $\sigma : [1, n] \to [1, n]$, let us call the weight of an edge as the absolute difference between the positions assigned to its endpoints in σ. Minimum Linear Arrangement is a well known NP-hard (Garey & Johnson, 1979) problem which consists of finding a permutation of the vertices of G such that the sum of the weights of its edges is minimized. More formally, we would like to find a permutation $\sigma : [1, n] \to [1, n]$ that minimizes $\sum_{\{v_{\sigma(i)}, v_{\sigma(j)}\} \in E} |j - i|$.

While Maximum Acyclic Subgraph deals (only) with the order of endpoints of arcs in the permutation, Minimum Linear Arrangement depends on their exact position. We now show that the decomposition property also allows to deal with such a problem. The idea is the following: consider $i < j$, and fix the subpermutation $\sigma(i, j)$, the sets $L_\sigma(i-1)$ of elements "on the left", and $R_\sigma(j+1)$ of elements "on the right". Then (see Figure 1 for an illustration of the contribution of edges):

- for an edge with both endpoints in $S_\sigma(i, j)$, we know its exact contribution (as $\sigma(i, j)$ is fixed). These edges have a global contribution $N_1 = \sum_{\{v_{\sigma(\ell)}, v_{\sigma(r)}\} \in E, i \leq \ell, r \leq j} |r - \ell|$. Note that N_1 depends only on $\sigma(i, j)$.

- For an edge with one endpoint in the left part $L_\sigma(i-1)$ and one endpoint in $S_\sigma(i, j)$, i.e., $\{v_{\sigma(\ell)}, v_{\sigma(r)}\}$ with $\ell < i$ and $i \leq r \leq j$, we will charge only for this edge the (yet partial) contribution $r - i$ (instead of $r - \ell$, as the exact left position is not fixed yet). Let N_2 be the sum of these contributions (note that N_2 depends only on $L_\sigma(i-1)$ and $\sigma(i, j)$).

- Similarly, for an edge with one endpoint in the right part $R_\sigma(j+1)$ and one endpoint in $S_\sigma(i, j)$, i.e., $\{v_{\sigma(\ell)}, v_{\sigma(r)}\}$ with $i \leq \ell \leq j$ and $j < r$, we will charge only for this edge the (yet partial) contribution $j - \ell$. Let N_3 be the sum of these contributions (note that N_3 depends only on $R_\sigma(j+1)$ and $\sigma(i, j)$).

We then define $g(i, j, L_\sigma(i-1), R_\sigma(j+1), \sigma(i, j)) = N_1 + N_2 + N_3$.

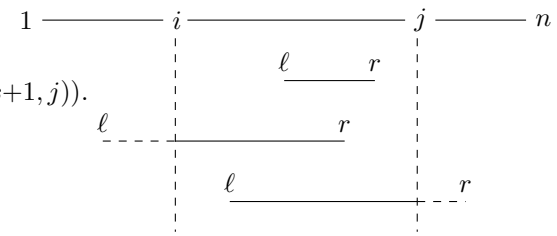

Figure 1. Contribution in g of edges, depending on the positions of their extremities. The contribution is in solid line. If $\ell < i$ and $r > j$ the contribution is 0.

It is clear that the first item of Definition 3.1 is satisfied (when $i = 1$ and $j = n$, N_1 equals the objective function, $N_2 = N_3 = 0$).

For the second item, let $i < s < j$. Consider an edge $\{v_{\sigma(\ell)}, v_{\sigma(r)}\}$. We show, for each possible case, how to recover the contribution of this edge to subproblem (i, j) (i.e., in $g(i, j, L_\sigma(i-1), R_\sigma(j+1), \sigma(i, j))$), see Figure 2 for an illustration:

- If both ℓ and r are in $[i, s]$, its contribution in $g(i, j, L_\sigma(i-1), R_\sigma(j+1), \sigma(i, j))$ is already counted in the "left call" $g(i, s, L_\sigma(i-1), R_\sigma(s+1), \sigma(i, s))$.

- Similarly, if both ℓ and r are in $[s+1, j]$, its contribution in $g(i, j, L_\sigma(i-1), R_\sigma(j+1), \sigma(i, j))$ is already counted in the "right call" $g(s+1, j, L_\sigma(s), R_\sigma(j+1), \sigma(s+1, j))$.

- If $i \leq \ell \leq s$ and $s+1 \leq r \leq j$, then the contribution is $(s - \ell)$ in the left call and $r - (s+1)$ in the right call, so in total $r - \ell - 1$. It only missed 1 to get the correct contribution $r - \ell$. So for these edges we have to add in total a contribution $C_1 = cut(S_\sigma(i, s), S_\sigma(s+1, j))$.

- If $\ell < i$ and $i \leq r \leq s$, then the contribution on the left call is $r - i$, the correct one. Similarly, if $s+1 \leq \ell \leq j$ and $j+1 \leq r$ the contribution of the right call is the correct one $(j - \ell)$.

- If $\ell < i$ and $s+1 \leq r \leq j$, the contribution in the left call is 0, the one in the right call is $r - (s+1)$. It misses $s + 1 - i$ to get the correct contribution $r - i$.

So for these edges we have to add in total $C_2 = (s + 1 - i) \cdot cut(L_\sigma(i-1), S_\sigma(s+1, j))$.

- Similarly, if $i \leq \ell \leq s$ and $r > j$, to get the correct charge we miss $C_3 = (j-s) \cdot cut(R_\sigma(j+1), S_\sigma(i, s))$.

Then, we define $h(L_\sigma(i-1), R_\sigma(j+1), S_\sigma(i, s), S_\sigma(s+1, j) = C_1 + C_2 + C_3$.

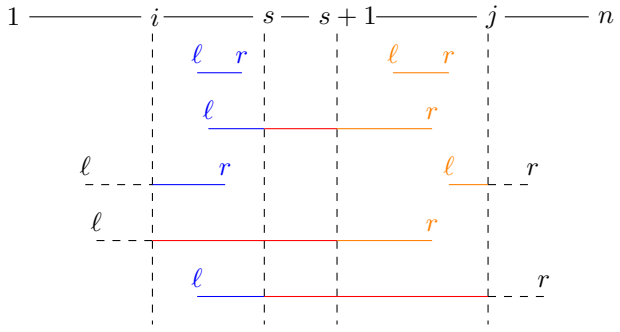

Figure 2. Decomposition in g: blue (resp. orange) corresponds to the contribution in the left call (resp. right call), red corresponds to missing contributions (that are counted in h). If $\ell < i$ and $r > j$ the contribution is 0.

Single-machine Sum of Completion Time Problem. We consider the following scheduling problem (denoted $1|prec|\sum C_j$ in the classical Graham's notation for scheduling problems), known to be strongly NP-hard (Lawler, 1978): we are given a set J of n jobs and a set of precedence constraints forming a DAG (J, A). The goal is to order the jobs, respecting precedence constraints, so as to minimize the sum of completion time of jobs.

Consider $i < j$, $\sigma(i, j)$ and $L_\sigma(i-1)$. With this information, we can compute the completion time of each job in $S_\sigma(i, j)$ (as we know the jobs before them ($L_\sigma(i-1)$) and the order $\sigma(i, j)$ of jobs in $S_\sigma(i, j)$). So we simply define g as the sum of these completion times (g depends on $S_\sigma(i, j)$ and $L_\sigma(i-1)$), with of course value ∞ if the order $\sigma(i, j)$ violates any constraints.

Item 1 of Definition 3.1 is trivially satisfied. For item 2, the left call computes the sum of completion times for jobs in $S_\sigma(i, s)$, and the right call the sum of completion times for jobs in $S_\sigma(s+1, j)$. Then h is ∞ if a constraint is violated (between a job whose position is between $s+1$ and j, and a job whose position is between i and s), and 0 otherwise.

Here again, this generalizes to many other single machine scheduling problems (involving other constraints and/or weights on jobs and/or other objective functions like sum of tardiness).

3.2. c-locality

The c-locality property states that the cost function of the problem depends only locally (with respect to the permutation/solution) on pairs of distinct elements. More formally, we have the following definition.

Definition 3.2. Let P be a permutation problem, with cost function f. We say that P is c-local if, on an instance I asking for a permutation σ on a set $A = \{a_1, \ldots, a_n\}$:

$$f_I(\sigma) = \sum_i cost_I(a_{\sigma(i-c)}, a_{\sigma(i-c+1)}, \ldots, a_{\sigma(i)}).$$

for some cost function[1] $cost_I$.

TSP. Given a complete graph on set V of vertices and a distance function $d(v_i, v_j)$ on pair of vertices, TSP asks to find a permutation σ that minimizes $\sum_{i=1}^{n} d(v_{\sigma(i)}, v_{\sigma(i+1)})$ (where $v_{\sigma(n+1)}$ is $v_{\sigma(1)}$). This NP-hard problem (Garey & Johnson, 1979) is then trivially 1-local (we can easily reformulate to get rid of the last term $d(v_{\sigma(n)}, v_{\sigma(1)})$).

Other routing problems can be shown to be 1-local as well.

Social Welfare Maximization in Keyword Auctions with Externalities. In standard Keyword Auctions (Edelman et al., 2007; Varian, 2007), a set $N = \{1, \ldots, n\}$ of advertisers compete over a set $K = \{1, \ldots, k\}$ of advertisement (or simply ad) slots, where $k \leq n$. Each player $i \in N$ has a valuation v_i per click and their ad has an intrinsic click probability $q_i \in (0, 1]$. Each slot $j \in K$ is associated with a click-through rate (ctr) λ_j, with $1 \geq \lambda_1 > \lambda_2 > \cdots \lambda_k > 0$. The overall ctr of an ad i in slot j is $\lambda_j q_i$. In the following, we assume that $k = n$, i.e., the number of slots k is equal to the number of ads n, for simplicity, by setting $\lambda_{k+1} = \cdots = \lambda_n = 0$ in case where $k < n$.

We aim to compute an assignment $\pi : N \to K$ of ads to slots (which for $k = n$ is a permutation of ads) so that the resulting expected *social welfare*, which is $\sum_{i=1}^{n} v_i \lambda_{\pi(i)} q_i$, is maximized.

In Keyword Auctions with Externalities (Gatti et al., 2018; Fotakis et al., 2011), the overall click-through rate of an ad i appearing in slot $\pi(i)$ also depends on the ads appearing in the c slots above $\pi(i) - c, \pi(i) - c + 1, \ldots, \pi(i) - 1$, where c is known as the *window size* and quantifies the scope of users' attention and memory when they process the ad list (Athey & Ellison, 2011). We let $Q_i(\pi)$ denote the i's ctr given the influence of the c ads above i in π. Then, we aim to compute an assignment $\pi : N \to [n]$ of ads to slots so that the resulting expected social welfare under externalities, which is $\sum_{i=1}^{n} v_i \lambda_{\pi(i)} Q_i(\pi)$, is maximized.

Social welfare maximization in keyword auctions with ex-

[1] To be more precise, to deal with the cases $i \leq c$ in the sum it should be $cost_I(a_{\sigma(t)}, a_{\sigma(t+1)}, \ldots, a_{\sigma(i)})$ for $t = \max\{i-c, 1\}$.

ternalities is NP-hard (Fotakis et al., 2011; Gatti et al., 2018) and is an example of a c-local permutation problem.

4. Proof of Theorem 1.1

In this section, we present the proof of Theorem 1.1, the main result of this paper. Using Lemma 2.6, what remains to be shown is that if a problem is decomposable or c-local, then it is polynomial time solvable when being $O(\log n)$-position enhanced. We provide a corresponding DP algorithm for decomposition in Lemma 4.1 (which extends the one of (Braverman & Mossel, 2008)) and for c-locality in Lemma 4.2

Lemma 4.1. *If a permutation optimization problem P is decomposable, then its k-position enhanced version is solvable in time $O(n \cdot 2^{O(k)} \cdot t)$ where t is the time to compute the value of a solution.*

This gives a polynomial computation time when $k = O(\log n)$, provided that $t \in poly(n)$.

Proof. We will use dynamic programming to find an optimal solution for the optimization problem P. Let σ^* be an (unknown) optimal permutation such that $|\sigma^*(i) - i| \leq c \log n$ for all i.

Let $i < j$ be any indices. Let $I^*(i,j)$ denote the elements in positions between i and j in σ, i.e.,

$$I^*(i,j) = \{a_{\sigma(i)}, a_{\sigma(i+1)}, \ldots, a_{\sigma(j)}\}.$$

Moreover, let

$$I^*_L(i) = \{a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(i-1)}\}$$

be the elements of the left of i and

$$I^*_R(j) = \{a_{\sigma(j+1)}, a_{\sigma(j+2)}, \ldots, a_{\sigma(n)}\}$$

the elements on the right of j. By assumption, we have $I^-_L(i) \subseteq I^*_L(i) \subseteq I^+_L(i)$, $I^-(i,j) \subseteq I^*(i,j) \subseteq I^+(i,j)$ and $I^-_R(j) \subseteq I^*_R(j) \subseteq I^+_R(j)$ where

$$I^+_L(i) = \{a_1, a_2, \ldots, a_{i+k-1}\},$$
$$I^-_L(i) = \{a_1, a_2, \ldots, a_{i-k-1}\},$$
$$I^+(i,j) = \{a_{i-k}, a_{i-k+1}, \ldots, a_{j+k}\},$$
$$I^-(i,j) = \{a_{i+k}, a_{i+k+1}, \ldots, a_{j-k}\},$$
$$I^+_R(j) = \{a_{j+1-k}, a_{j+2-k}, \ldots, a_n\},$$
$$I^-_R(j) = \{a_{j+1+k}, a_{j+2+k}, \ldots, a_n\}.$$

Thus, selecting the sets $I^*_L(i)$, $I^*(i,j)$ and $I^*_R(j)$ involves selecting k elements from a list of $2k$ elements for $I^*_L(i)$ and $2k$ elements from $4k$ elements for $I^*(i,j)$. Once $I^*_L(i)$, $I^*(i,j)$ are fixed, the remaining elements belong to $I^*_R(j)$ (see Figure 3 for an illustration). Thus the number of different guesses we have to do is bounded by $2^{2k+4k} = 2^{6k}$.

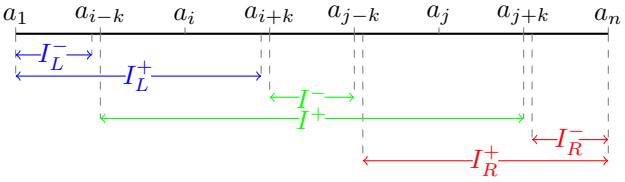

Figure 3. Selecting the set I^* involves choosing $j - i + 1$ elements that include all elements of I^- and are contained within I^+ (highlighted in green). This corresponds to selecting $2k$ elements from the list $\{a_{i-k}, a_{i-k+1}, \ldots, a_{i+k-1}, a_{j-k+1}, \ldots, a_{j+k}\}$ of $4k$ elements. Similarly, selecting I^*_L requires choosing $i - 1$ elements that include I^-_L and are contained within I^+_L (in blue). This involves selecting k elements from the list $\{a_{i-k}, \ldots, a_{i+k-1}\}$ of $2k$ elements. The same logic applies for I^*_R (in red).

Now, let us define for any i, j the set $\mathcal{S}(i,j)$ of sets $I'(i,j)$ of size $j - i + 1$ such that $I^-(i,j) \subseteq I'(i,j) \subseteq I^+(i,j)$. We have that $I^*(i,j) \in \mathcal{S}(i,j)$.

Moreover, let $\mathcal{L}(i-1)$ be the set of sets $I'_L(i)$ of size $i - 1$ such that $I^-_L(i) \subseteq I'_L(i) \subseteq I^+_L(i)$. We have that $I^*_L(i) \in \mathcal{L}(i-1)$. Similarly, we define the set $\mathcal{R}(j+1)$ of sets $I'_R(j)$ of size $n - j$.

We now define the following problem $\mathcal{I}(i,j)$: for each $I'(i,j) \in \mathcal{S}(i,j)$, each $I'_L(i) \in \mathcal{L}(i-1)$ and each $I'_R(j) \in \mathcal{R}(j+1)$ such that $I'(i,j)$, $I'_L(i)$ and $I'_R(j)$ are pairwise disjoint, find a permutation σ' of elements in $I'(i,j)$ such that:

- $|\sigma'(t) - t| \leq k$,
- $\{a_{\sigma'(i)}, \ldots, a_{\sigma'(j)}\} = I'(i,j)$
- $\sigma'(i,j)$ optimizes the function g given in the decomposition property of P.

Based on the decomposition property of g, we will solve the problem via dynamic programming. Note that we find an optimal solution by solving $\mathcal{I}(1,n)$. Assume for the sake of simplicity that n is a power of two. We will solve $\mathcal{I}(1,n)$ using the solutions of $\mathcal{I}(1,n/2)$ and $\mathcal{I}(n/2+1,n)$, and so on. We solve the leaves of the tree (containing only one element) in constant time, and we have it total $n - 1$ subproblems.

Let us explain how we solve $\mathcal{I}(i,j)$ using the solutions of $\mathcal{I}(i,s)$ and $\mathcal{I}(s+1,j)$, where $i \leq s \leq j$.

Let $I'(i,j) \in \mathcal{S}(i,j)$, $I'_L(i) \in \mathcal{L}(i-1)$ and $I'_R(j) \in \mathcal{R}(j+1)$. Thanks to the decomposition property on g

(see Definition 3.1), $I'(i,j)$, $I'_L(i)$ and $I'_R(j)$ being fixed, an optimal $\sigma'(i,j)$ is composed of an optimal on $[i,s]$ and an optimal solution on $[s+1,j]$. There are at most $2^{O(k)}$ choices to partition $I'(i,j)$ into two sets $I'(i,i+s) \in \mathcal{S}(i,s)$ and $I'(s+1,j) \in \mathcal{S}(s+1,j)$. So we can compute an optimal solution of our subproblem using $2^{O(k)}$ calls to the DP table (on subproblems in $\mathcal{I}(i,s)$ and $\mathcal{I}(s+1,j)$). □

Now we can also prove a similar lemma for the case of a c-local permutation optimization problem, using a different DP.

Lemma 4.2. *Let P be a c-local permutation problem. Its k-position enhanced version is solvable in time $O(n \cdot 2^{2k} \cdot k^{c+1})$.*

Proof. We will use again dynamic programming to find an optimal solution for problem P. Let i be any index. Let $I^*(i)$ denote the elements in an optimal order in positions between 1 and i in σ, i.e., $I^*(i) = \{a_{\sigma(1)}, \ldots, a_{\sigma(i)}\}$.

By assumption, we have that $I^-(i) \subseteq I^*(i) \subseteq I^+(i)$, where

$$I^-(i) = \{a_1, a_2, \ldots, a_{i-k}\}, \quad I^+(i) = \{a_1, a_2, \ldots, a_{i+k}\}.$$

Thus, selecting the set $I^*(i)$ involves selecting k elements from a list of $2k$ elements. Therefore, the number of different guesses we have to do is bounded by 2^{2k}.

Let us now define for any i the set $S_\sigma(i)$ of sets $I'(i)$ of size i such that $I^-(i) \subseteq I'(i) \subseteq I^+(i)$. We have that $I^*(i) \in S_\sigma(i)$.

Similarly, we let $I^*(i+1, i+c)$ denote the elements in positions between $i+1$ and $i+c$ in σ. Here, we have to select each of these c elements among a list of $2k+1$ elements, so the number of different guesses is bounded by $O(k^c)$ (as c is a constant). Moreover, for each guess we also consider all different permutations of the c elements (denoted by $\sigma(i, i+c)$) which takes constant time (as c is constant).

We also define for any i the set $\mathcal{S}(i+1, i+c)$ of sets $I'(i+1, i+c)$ of size c such that $I^-(i+1, i+c) \subseteq I'(i+1, i+c) \subseteq I^+(i+1, i+c)$. We have that $I^*(i+1, i+c) \in \mathcal{S}(i+1, i+c)$.

Let us now explain how we solve the problem using dynamic programming which is based on the property of c-locality. For every entry of the DP table we have the following:

$$DP(S_\sigma(i), a_{\sigma(i+1)}, \ldots, a_{\sigma(i+c)}) = \min_{\sigma(i)} \big\{ DP(S_\sigma(i) \setminus \{\sigma(i)\}, a_{\sigma(i)}, \ldots, a_{\sigma(i+c-1)}) + cost_I(a_{\sigma(i)}, a_{\sigma(i+1)}, \ldots, a_{\sigma(i+c)}) \big\}.$$

There are at most $2k+1$ choices for $a_{\sigma(i)}$, so each DP entry can be computed in time $O(k)$. Overall, we have that the running time is $O(n \cdot 2^{2k} \cdot k^{c+1})$. □

5. About position-enhanced permutation problems

The dynamic programming algorithms of Lemmas 4.1 and 4.2 show that $O(\log n)$-position enhanced versions of decomposable or c-local permutation problems are solvable in polynomial time. In this section, we show two complementary results on position-enhanced versions of permutation problems that, though technically quite easy, enlight interesting limitations to Lemma 2.6:

- First we show that there are some permutation problems which remain hard to solve in polynomial time even when being $O(\log n)$-position enhanced.
- Second, we show that this bound $O(\log n)$ is tight, meaning that there are decomposable problems and c-local problems which remain hard when being $f(n)\log n$-position enhanced, as soon as f is unbounded.

5.1. A $\log(n)$-position-enhanced hard problem

Let us consider the following problem called Permutation Clique: we are given a graph $G = (V, E)$ on $n = t^2$ vertices $v_{i,j}$, $i, j = 1, \ldots, t$. The goal is to determine whether there exists a permutation σ over $\{1, \ldots, t\}$ such that the t vertices $v_{i,\sigma(i)}$, $i = 1, \ldots, t$, form a clique in G. If one puts vertices in a tabular of size $t \times t$, then we want to find a clique of t vertices with exactly one vertex per row and one per column. This problem is trivially solvable in time $2^{O(t \log t)}$. Interestingly, (Lokshtanov et al., 2018) showed the following, where ETH stands for Exponential Time Hypothesis:

Proposition 5.1. *(Lokshtanov et al., 2018) Under ETH, Permutation Clique is not solvable in time $2^{o(t \log t)}$.*

We show that this problem remains hard even when being $O(\log t)$-position enhanced. Formally, the problem is defined as follows. We are given an instance $G = (V, E)$ of Permutation Clique on t^2 vertices, and we want to determine if there is a permutation σ over $\{1, \ldots, t\}$ such that:

(1) $|\sigma(i) - i| \leq c \log t$ for all $i = 1, \ldots, t$;

(2) $\{v_{i,\sigma(i)} : i = 1, \ldots, t\}$ is a clique in G.

Proposition 5.2. *Under ETH, $(c \log t)$-Positioned-Enhanced Permutation Clique is not solvable in polynomial time (for any $c > 0$).*

Proof. Let us consider an instance $G = (V, E)$ of Permutation Clique, on $n = t^2$ vertices $\{v_{i,j} : i, j = 1, \ldots, t\}$. Let $t' = 2^{t/c}$. We build from G a graph $G' = (V', E')$ on $n' = t'^2$ vertices $\{v'_{i,j} : i, j = 1, \ldots, t'\}$ where:

- For $i_1, i_2, j_1, j_2 \leq t$: v'_{i_1,j_1} is adjacent to v'_{i_2,j_2} if and only if v_{i_1,j_1} is adjacent to v_{i_2,j_2} in G;
- For $i, j \leq t$ and $\ell > t$: $v'_{i,j}$ is adjacent to $v_{\ell,\ell}$;
- For $\ell_1, \ell_2 > t$: v_{ℓ_1,ℓ_1} is adjacent to v_{ℓ_2,ℓ_2}.

By construction, a (non trivial) clique in G' is composed by a clique in G plus 'diagonal' vertices $v_{\ell,\ell}$. Hence, there is a permutation clique of size t' in G' if and only if there is a permutation clique of size t in G. Moreover, we know by construction that if there is a permutation clique induced by σ' in G', then for $\ell > t$ $\sigma(\ell) = \ell$ (diagonal vertices), and for $\ell \leq t$ $\sigma(\ell) \leq t$. Then in any case, $|\sigma(\ell) - \ell| \leq t = c \log t'$.

Suppose that we answer to $(c \log t')$-Positioned-Enhanced Permutation Clique in polynomial time $O(n'^c)$. The construction of G' takes time $O(n')$, so we can solve Permutation Clique in time $O(n'^c) = O(t'^{c'}) = 2^{O(t)}$. This is in contradiction with ETH. □

5.2. Hardness results for worse than $O(\log n)$-position-enhancement

Let us now go back to decomposable and c-local problems, and show that the $O(\log n)$-position enhancement is necessary for some problems.

Proposition 5.3. *For any unbounded increasing function f, $f(n) \log n$-Positioned-Enhanced Max Acyclic Subgraph is not solvable in polynomial time under ETH.*

Proof. We make a reduction from Max Acyclic Subgraph, which is not solvable in time $2^{o(n)}$ under ETH (from the linear reduction from vertex cover (Karp, 1972) and the hardness of vertex cover (Impagliazzo et al., 2001)). Let $G = (V, E)$ be a directed graph. We build a graph G' by adding to G dummy vertices so that G' has $N = 2^{n/f(n)}$ vertices. We keep the arcs that are in G but do not add any new arcs. Then the order in which we put dummy vertices does not change the objective function. So we know that there is an optimal solution such that $\sigma(i) = i$ for $i > n$ and $\sigma(i) \leq n$ for $i \leq n$, and there σ restricted to $\{1, \ldots, n\}$ is an optimal solution for G. So we know that $|\sigma(i) - i| \leq n = f(n) \log N \leq f(N) \log N$.

Now, suppose that we can solve $f(n) \log n$-Positioned-Enhanced Max Acyclic Subgraph in polynomial time $O(N^c)$. Then we would solve Max Acyclic Subgraph in time $O(2^{cn/f(n)}) = 2^{o(n)}$ as f is unbounded. This is impossible under ETH. □

Proposition 5.4. *For any unbounded increasing function f, $f(n) \log n$-Positioned-Enhanced TSP is not solvable in polynomial time under ETH.*

Proof. The proof is similar to the one of Proposition 5.3. This time, we add dummy vertices that are all at distance 0 from a given initial vertex v (and all distances between them are 0), so that there are $N = 2^{n/f(n)}$ vertices in total. Then we know that there exists an optimal solution in which these vertices are ranked last in the permutation (the solution starts with v), so we get as previously $f(N) \log N$-position enhancement "for free". We get the same conclusion, using the fact that TSP is not solvable in time $2^{o(n)}$ under ETH (Impagliazzo et al., 2001). □

Similar proofs can be easily derived for other problems, such as Min Linear Arrangement and the single-machine sum of completion time problem considered in Section 3.

Acknowledgement

Dimitris Fotakis has been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program. Michalis Xefteris was supported by the framework of H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union – NextGenerationEU (H.F.R.I. Project Number: 15877). Panagiotis Patsilinakos has received support under the program "Investissements d'Avenir" launched by the French Government,project PSL JRC 2024 - N 2024-488.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning by further exploring its interactions with Theoretical Computer Science. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Antoniadis, A., Eliáš, M., Polak, A., and Venzin, M. Approximation algorithms for combinatorial optimization with predictions, 2024. arXiv:2411.16600.

Athey, S. and Ellison, G. Position Auctions with Consumer Search. *The Quarterly Journal of Economics*, 126(3): 1213–1270, 08 2011.

Bampis, E., Escoffier, B., and Xefteris, M. Parsimonious learning-augmented approximations for dense instances

of $\mathcal{NP}$-hard problems. In *Proc. 41st Int. Conf. Machine Learning (ICML)*, volume 235, pp. 2700–2714, 2024.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2):405–421, 2021.

Braverman, M. and Mossel, E. Noisy sorting without resampling. In *Proc. 19th Symp. Discret. Algorithms (SODA)*, pp. 268–276, 2008.

Braverman, M. and Mossel, E. Sorting from noisy information, 2009. arXiv:0910.1191.

Braverman, V., Dharangutte, P., Shah, V., and Wang, C. Learning-Augmented Maximum Independent Set. In *Approx., Random., and Comb. Optim. Algorithms and Techniques (APPROX/RANDOM)*, volume 317, pp. 24:1–24:18, 2024.

Busa-Fekete, R., Fotakis, D., Szörényi, B., and Zampetakis, M. Optimal learning of mallows block model. In *Conference on Learning Theory, COLT 2019*, volume 99 of *Proceedings of Machine Learning Research*, pp. 529–532. PMLR, 2019.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Velickovic, P. Combinatorial optimization and reasoning with graph neural networks. *J. Mach. Learn. Res.*, 24:130:1–130:61, 2023.

Chen, J. Y., Silwal, S., Vakilian, A., and Zhang, F. Faster fundamental graph algorithms via learned predictions. In *International Conference on Machine Learning, ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3583–3602. PMLR, 2022.

Cohen-Addad, V., d'Orsi, T., Gupta, A., Lee, E., and Panigrahi, D. Learning-augmented approximation algorithms for maximum cut and related problems. In *Proc. 38th Adv. Neural Inf. Processing Syst. (NeurIPS)*, volume 37, pp. 25961–25980, 2024.

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Faster matchings via learned duals. In *Proc. 35th Adv. Neural Inf. Processing Syst. (NeurIPS)*, 2021.

Dong, Y., Peng, P., and Vakilian, A. Learning-Augmented Streaming Algorithms for Approximating MAX-CUT. In *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325, pp. 44:1–44:24, 2025.

Edelman, B., Ostrovsky, M., and Schwarz, M. Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords. *American Economic Review*, 97(1):242–259, March 2007.

Ergun, J. C., Feng, Z., Silwal, S., Woodruff, D., and Zhou, S. Learning-augmented k-means clustering. In *Proc. 10th Int. Conf. Learning Representations (ICLR)*, 2022.

Fotakis, D., Krysta, P., and Telelis, O. Externalities among advertisers in sponsored search. In *Proc. 4th Int. Symp. on Algorithmic Game Theory (SAGT)*, volume 6982, pp. 105–116, 2011.

Gamlath, B., Lattanzi, S., Norouzi-Fard, A., and Svensson, O. Approximate cluster recovery from noisy labels. In *Proc. 35th Conf. on Learning Theory (COLT)*, 2022.

Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN 0-7167-1044-7.

Gatti, N., Rocco, M., Serafino, P., and Ventre, C. Towards better models of externalities in sponsored search auctions. *Theoretical Computer Science*, 745:150–162, 2018.

Ghoshal, S., Makarychev, K., and Makarychev, Y. Constraint satisfaction problems with advice, 2024. arXiv:2403.02212.

Im, S., Kumar, R., Petety, A., and Purohit, M. Parsimonious learning-augmented caching. In *Proc. 39th Int. Conf. Machine Learning (ICML)*, volume 162, pp. 9588–9601, 2022.

Impagliazzo, R., Paturi, R., and Zane, F. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

Karp, R. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103. Plenum Press, 1972.

Khalil, E. B., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 6348–6358, 2017.

Lawler, E. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Ann. Discrete Math.*, 2:75–90, 1978.

Lindermayr, A. and Megow, N. ALPS. https://algorithms-with-predictions.github.io/.

Lokshtanov, D., Marx, D., and Saurabh, S. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. *J. ACM*, 68(4), 2021.

Mahdian, M., Nazerzadeh, H., and Saberi, A. Allocating online advertisement space with unreliable estimates. In *Proc. 8th ACM Conf. on Electronic Commerce (EC)*, pp. 288–294, New York, NY, USA, 2007.

Nguyen, T. D., Chaturvedi, A., and Nguyen, H. Improved learning-augmented algorithms for k-means and k-medians clustering. In *Proc. 11th Int. Conf. Learning Representations (ICLR)*, 2023.

Sakaue, S. and Oki, T. Discrete-convex-analysis-based framework for warm-starting algorithms with predictions. In *Proc. 36th Adv. Neural Inf. Processing Syst. (NeurIPS)*, 2022.

Varian, H. R. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

Online Fair Division for Personalized 2-Value Instances

Georgios Amanatidis[1,2], Alexandros Lolos[1,2], Evangelos Markakis[1,2,3], and Victor Turmel[4]

[1]Department of Informatics, Athens University of Economics and Business, Athens, Greece.
[2]Archimedes, Athena Research Center, Athens, Greece.
[3]Input Output Global (IOG), Athens, Greece.
[4]Institut de Mathématique, Université Paris-Saclay, Orsay, France.

Abstract

We study an online fair division setting, where goods arrive one at a time and there is a fixed set of n agents, each of whom has an additive valuation function over the goods. Once a good appears, the value each agent has for it is revealed and it must be allocated immediately and irrevocably to one of the agents. It is known that without any assumptions about the values being severely restricted or coming from a distribution, very strong impossibility results hold in this setting [He et al., 2019, Zhou et al., 2023]. To bypass the latter, we turn our attention to instances where the valuation functions are restricted. In particular, we study *personalized* 2*-value instances*, where there are only two possible values each agent may have for each good, possibly different across agents, and we show how to obtain worst case guarantees with respect to well-known fairness notions, such as *maximin share fairness* and *envy-freeness up to one (or two) good(s)*. We suggest a deterministic algorithm that maintains a $1/(2n-1)$-MMS allocation at every time step and show that this is the best possible any deterministic algorithm can achieve if one cares about *every single* time step; nevertheless, eventually the allocation constructed by our algorithm becomes a $1/4$-MMS allocation. To achieve this, the algorithm implicitly maintains a fragile system of priority levels for all agents. Further, we show that, by allowing some limited access to future information, it is possible to have stronger results with less involved approaches. In particular, by knowing the values of goods for $n-1$ time steps into the future, we design a matching-based algorithm that achieves an EF1 allocation every n time steps, while always maintaining an EF2 allocation. Finally, we show that our results allow us to get the first nontrivial guarantees for additive instances in which the ratio of the maximum over the minimum value an agent has for a good is bounded.

1 Introduction

The problem of fairly allocating a set of resources to a set of agents without monetary exchanges was mathematically formalized only in the 1940's by Steinhaus [1949]—along with his students Banach and Knaster. Nevertheless, fair division is such a fundamental concept that the celebrated *Cut & Choose* protocol already appears in ancient literature. For this reason, since the formal introduction of the problem, numerous variants, each under different assumptions and constraints, have been studied in mathematics, economics, political science and, more recently, in computer science. Specifically, as far as the latter is concerned, the algorithmic nature of most questions one may ask about fair division problems has lead to a flourishing literature on the topic, often on variants that deal with discrete, indivisible resources.

In the most standard variants of the problem a set of resources and a set of agents, each equipped with a valuation function, are given as an input and one would like to produce a, complete or partial, partition of the resources to the agents, so that some predetermined *fairness criterion* is satisfied. Here we study a version of the problem where the set of n agents is indeed given, but the indivisible goods arrive in an *online* fashion. That is, at each time step a new good appears, its value for the agents is revealed, and the good must be irrevocably assigned to a single agent before the next good arrives. Online fair division problems of similar nature appear in many real-world scenarios, like, e.g., the operation of food banks, donation distribution in disaster relief, limited resource allocation within an organization like a hospital or a university, or memory and CPU management in cloud computing.

Despite their wide applicability, however, online fair division settings have not been studied nearly as much as their offline counterparts. And admittedly, there is a good reason for the relative scarcity of such works. Namely, it is relatively easy to show strong impossibility worst-case results, even if one aims for rather modest fairness guarantees. For example, when the agents have additive valuation functions and the items are goods (i.e., they do not have a negative value for any agent) that arrive adversarially—as is the case here—there is no deterministic algorithm that achieves *any* positive approximation factor with respect to *maximin share fairness* (MMS) [Zhou et al., 2023] or to *envy-freeness up to one good* (EF1) [He et al., 2019]. Even if one allows some item reallocations in order to fix that, the number of reallocations cannot be bounded by any function of n [He et al., 2019].

Of course, there are ways to bypass these results, like assuming that the item values are drawn from distributions [Benade et al., 2018, Zeng and Psomas, 2020], restricting the valuation functions [Aleksandrov et al., 2015] or the number of distinct agent types [Kulkarni et al., 2025], allowing items to be reallocated [He et al., 2019], relaxing the requirement for fairness guarantees at the intermediate time steps [Cookson et al., 2025], augmenting the input with predictions [Banerjee et al., 2022] or even with full knowledge of the whole instance [Elkind et al., 2024, Cookson et al., 2025]. Here the main approach we take is to focus on designing *deterministic* algorithms for instances with *restricted* valuation functions, although we also explore how information about the future allows for stronger results with simpler algorithms. In particular, we mostly consider instances where each agent i has two values for the goods: a low value β_i and a high value α_i. These instances, which we call *personalized* 2*-value instances*, generalize the binary case and the setting where there are only two types of goods; until now, these were the only cases where positive worst-case results were known (by Aleksandrov et al. [2015] and Elkind et al. [2024], respectively). Consequently, personalized 2-value instances capture the natural dichotomy between highly desirable and less desirable goods in a more nuanced way which also varies across agents. Moreover, there is a natural way of approximating any additive instance via a 2-value instance by just setting a threshold for each agent i and rounding everything up (to the "high value" α_i) or down (to the "low value" β_i) accordingly. Although this approximation is not always meaningful, it does give nontrivial guarantees for agents whose values are bounded by lower and upper bounds that do not differ by too much, as we argue in Section 6.

It should be noted that this type of restriction has drawn significant attention in the fair division literature recently. The study of k-*value instances* (often called k-*valued* or, in the case where $k = 2$, *bi-valued*), where there are at most k possible common values that can represent the value an agent has for a good, was introduced by Amanatidis et al. [2021] as a way to study the existence of EFX allocations under a restriction that made the question easier, yet not straightforward. Indeed, it turns out that, in many cases, k-value instances seem to strike a good balance between maintaining the flavor and many of the challenges of the general additive version of the problem and allowing nontrivial positive results, even for $k = 2$ or 3. Consequently, there is a recent line of work studying fair division questions under such restrictions (e.g., [Akrami et al., 2022, Garg et al., 2022, Aziz et al., 2023, Amanatidis et al., 2024, Fitzsimmons et al., 2024]), often allowing for the k values to be different per agent (hence, *personalized*), first studied by Murhekar and Garg [2021] for general k, under the name k-*ary instances*.

Finally, as we mentioned above, we also explore another relaxation of the problem which is somewhat orthogonal to that of restricting the valuation functions, and aims to alleviate the lack of information due to the online arrival of the goods. Several recent works augment their online algorithms with additional information about the future in the form of (possibly erroneous) predictions [Lykouris and Vassilvitskii, 2021, Banerjee et al., 2022, 2023, Balkanski et al., 2024, Benomar and Perchet, 2024]. Here we follow an approach closer to He et al. [2019], Cookson et al. [2025], and Elkind et al. [2024], who assume that the whole instance can be known in advance and what truly remains online is the allocation process itself. As the number of goods could be much larger than the number of agents in many scenarios, we feel that knowing the whole instance is a rather strong assumption; instead, we allow some of our algorithms to see into the future only for a number of time steps that is comparable to the number of agents. In our setting this turns out to be enough for getting simpler algorithms with solid guarantees.

Contribution and Technical Considerations. Following the aforementioned line of work on restricting the valuation functions, we explore what is possible for (personalized) 2-value instances in deterministic online fair division. We obtain positive and negative results, which we then extend beyond our main setting. Specifically:

- The impossibility results one has for general additive instances persist here as well, albeit milder. We show that, for any $\varepsilon > 0$, no algorithm can guarantee $(1/2 + \varepsilon)$-EF1 at every time step, even for two agents (Theorem 3.1), or $(1/(2n-1) + \varepsilon)$-MMS at every time step for general n (Theorem 3.2).
- We present an algorithm with tight approximation guarantee with respect to MMS. In particular, our Algorithm 1 guarantees $1/(2n-1)$-MMS at every time step, which improves to $\Omega(1)$ at every time step, assuming that m is large enough (Theorem 4.1 and Corollary 4.5).
- We demonstrate that even very limited knowledge of the future may help significantly, as our Algorithm 2 guarantees EF1 at every *other* time step for two agents just by looking one step ahead into the future (Theorem 5.2).
- More generally, we show that having a foresight of $n-1$ goods into the future suffices in order to guarantee EF2 at every time step and EF1 at every n time steps for n agents (Algorithm 3 and Theorem 5.4). The latter also implies a $1/n$-MMS approximation at every n time steps, whereas achieving $(1/n + \varepsilon)$-MMS at every time step is impossible, even if the whole instance is fully known in advance.
- We provide a simple reduction that allows us to translate our results to general additive instances at the expense of a multiplicative factor that depends on the largest ratio between the values of any two goods. To the best of our knowledge, these are the first positive results in this setting (Theorem 6.1 and Corollaries 6.4 and 6.5).

While the main ideas in all of our results are very intuitive at a high level, turning them into tight or best-possible statements is far from trivial. Our impossibility results (especially Theorem 3.2) cannot rely on boosting values as needed (as it is commonly done in the literature), given the nature of the valuation functions we study. Instead, we need a family of instances fully adjusted to what *any* algorithm could do in order to determine the allocation with enough precision to get what turns out to be a tight bound. Algorithm 1, despite its several cases and careful book-keeping, is based on the simple idea that throughout the allocation process an agent should gain higher priority the more value they lose to others. Although our parametrization of the indices that imply this priority (one out of every $2n-1$ goods in general; one out of $3n-2$ high-valued goods) may seem rather loose, not only is it tight but it requires an involved analysis that includes distinct inductive proofs (that differ drastically) for the initial and later phases of the algorithm, respectively. Algorithms 2 and 3 are simpler, especially the former, but there are subtleties here as well. Ideally, what we would like to achieve between time steps kn and $(k+1)n$ would be to get an EF1 allocation by taking the union of two EF1 allocations. However, in general, this fails trivially, even for agents with binary valuation functions. Instead, we take advantage of the structure of our instances and built the second EF1 allocation (the one with the current and predicted goods) so that its envy graph "cancels out" the problematic edges of the envy graph of the current allocation, via carefully defined auxiliary valuation functions.

Further Related Work. There is a vast literature on fair division, both with divisible and with indivisible resources. For a recent survey on the latter, see Amanatidis et al. [2023]. Here we focus on online fair division settings, mostly with indivisible items.

Aleksandrov et al. [2015] introduced the setting we study here. Their work focused more on binary instances and on welfare guarantees. Subsequent works studied the mechanism design version of the problem mostly for binary instances [Aleksandrov and Walsh, 2017] and explored the limitations of achieving fairness notions like EF1 with general additive or even with non-additive binary valuations functions [Aleksandrov and Walsh, 2019]. The results of these early works are summarized in the survey of Aleksandrov and Walsh [2020]. A viable direction in order to bypass the existing strong negative results is to assume that there are underlying distributions with a bounded support. In such a setting Benade et al. [2018] show that it is possible to achieve envy that grows sublinearly with time with a very simple algorithm and that this is asymptotically best possible, whereas Zeng and Psomas [2020] study the compatibility of efficiency and envy in variants of the setting. However, the works that are closer to ours, one way or another, are those of He et al. [2019], Zhou et al. [2023], Cookson et al. [2025], and Elkind et al. [2024].

He et al. [2019] show that it is impossible to achieve temporal-EF1 (or even any nontrivial approximation of it) unless a linear fraction of the goods are reallocated. Then they design algorithms (occasionally augmented with full knowledge of the future) that achieve temporal-EF1 with a bounded number of reallocations. Zhou et al. [2023] focus on temporal-MMS and show that no approximation is possible for goods beyond the case of two agents. Then they turn their attention to chores (i.e., items that no agent values positively) where they present an algorithm with constant approximation guarantee. It should be noted here that although their 0.5-temporal-MMS algorithm for $n=2$ seems to contradict our Theorem 3.2, this is made possible by the additional assumption that $v_i(M)$ is known for all $i \in N$, which we do not make here. The very recent works of Elkind et al. [2024] and Cookson et al. [2025] formalize the notion of *temporal fairness* (although in slightly different ways) and assume that the full information about an instance is known upfront. However, even under this strong assumption, there still are impossibility results and, hence, both papers mostly focus on further restrictions. Cookson et al. [2025] obtain positive results at every step of the allocation for the case of two agents, and for the overall allocation for instances where the agents agree on the ordering over the goods or for the variant of the problem where the same set of goods arrives at every time step.

Elkind et al. [2024], on the other hand, show temporal EF1 guarantees for two agents, two types of items, generalized binary valuations, and for unimodal preferences.

There is also a line of work where the setting is very similar to ours but the items that arrive online are divisible [Gkatzelis et al., 2021, Barman et al., 2022, Banerjee et al., 2022, 2023]. The similarities, however, are only superficial, as the fractional assignment of even a few goods allows us to bypass most strong impossibility results. Finally, online fair division with indivisible items has been very recently studied in the context of bandit learning [Yamada et al., 2024, Procaccia et al., 2024].

Of course, besides the online setting where the items arrive over time, a different scenario is to have a known set of resources and assume that agents arrive in an online fashion. Indeed, there is a significant amount of work in this direction, with an emphasis on indivisible resources [Kalinowski et al., 2013, Kash et al., 2014, Sinclair et al., 2021, Vardi et al., 2022, Banerjee et al., 2024]. For indivisible resources, the very recent work of Kulkarni et al. [2025] obtains the first solid maximin share fairness guarantees for this setting assuming that the agents fit into a limited number of different types. Nevertheless, these settings have very different flavor than ours.

2 Preliminaries

Let $N = [n] = \{1, 2, \ldots, n\}$ be set of agents and $M = \{g_1, g_2, \ldots, g_m\}$ be a set of indivisible goods, where $n, m \in \mathbb{N}$. The high-level goal is to assign the goods of M to the agents of N in a way that is considered fair according to a well-defined fairness criterion. We usually call this assignment of goods to agents an *allocation*, which can be *partial* (if not all goods of M have been assigned) or complete (i.e., it defines a partition of M). Formally, an allocation $(A_1, \ldots, A_n)$ is an ordered tuple of disjoint subsets of M; we often call the set A_i the *bundle* of agent i.

Each agent i is associated with an *additive* set function $v_i : 2^M \to \mathbb{R}_{\geq 0}$, where $v_i(S) = \sum_{g \in S} v_i(\{g\})$ represents the value of agent i for the subset S of goods. When S is a singleton, we usually write $v_i(g)$ rather than $v_i(\{g\})$. Of course, valuation functions of the agents can be more general but in this work we only study special cases of additive instances of the problem, i.e., instances where all agents have additive valuation functions that are restricted in some way. In particular, we focus on *personalized* 2*-value* and *personalized interval-restricted* instances.

Definition 2.1 (Personalized 2-Value Instances)**.** We say that an instance of the problem is a *personalized* 2*-value instance*, if for any $i \in N$ the function v_i is additive and there are $\alpha_i \geq \beta_i \geq 0$, such that for any $g \in M$, it holds that $v_i(g) \in \{\alpha_i, \beta_i\}$. When $\alpha_i = \alpha$, $\beta_i = \beta$, for all $i \in N$, we call this a 2*-value instance*.

One could reasonably claim here that the interesting case is when $\alpha_i > \beta_i > 0$ in Definition 2.1; indeed we say that such agents are of *type 1*. Similarly, if $\alpha_i = \beta_i > 0$, agent i is of *type 2* and if $\alpha_i > \beta_i = 0$, agent i is of *type 3*. The remaining agents (i.e., $\alpha_i = \beta_i = 0$) are of *type 0* and are completely irrelevant, as they are trivially satisfied (namely, they see the allocation as being temporal-EF) by an empty bundle. So, without loss of generality, we may assume that the instances we consider only contain agents of types 1, 2, and 3. Of course, agents of type 2 and 3 are easier to satisfy (see, e.g., Corollary 4.4). As a final observation about agent types, we note that for the fairness notions we introduce below and study in this work, scaling the valuation functions has no effect on the fairness guarantees of any given allocation. Thus, it is also without loss of generality, to assume that $\beta_i = 1$ if agent i is of type 1 or 2, or that $\alpha_i = 1$ if it is of type 3.

The next definition aims to capture the continuous analog of 2-value instances, i.e., we would like all the values agent i may have for a good to be in the interval $[\beta_i, \alpha_i]$. However, as we just mentioned above, scaling the valuation functions is irrelevant for the fairness notions we consider. That is, any general additive instance can be trivially transformed to an equivalent instance where, for any $i \in N$ and $g \in M$,

it holds that $v_i(g) \in [0, 1]$. Thus, in order for the restriction to be meaningful in our context, it should hold that $\beta_i > 0$ for all $i \in N$. By scaling appropriately, this is equivalent to asking that $\beta_i = 1$ for all $i \in N$.

Definition 2.2 (Personalized Interval-Restricted Instances)**.** We say that an instance of the problem is a *personalized interval-restricted instance*, if for any $i \in N$ the function v_i is additive and there is $\alpha_i > 1$, such that for any $g \in M$, it holds that $v_i(g) \in [1, \alpha_i]$. When $\alpha_i = \alpha$, for all $i \in N$, we call this an *interval-restricted instance*.

By maintaining a bound on $\max_{i \in [n]} \sqrt{\alpha_i}$, i.e., the ratio of the highest over the lowest value an agent has for a good in Definition 2.2, we can reduce the problem of dealing with personalized interval-restricted instances to dealing with personalized 2-value proxy instances instead (albeit at the expense of the approximation ratio guarantees; see Section 6).

In standard—offline—fair division settings, all the goods of M are known and available to be used as an input to an allocation algorithm. Here we consider an *online* fair division setting, where the goods arrive one at a time; the set M (or even its cardinality, m) is not known a priori. When a good g arrives, its value for each agent is revealed and it needs to be added to the bundle of some agent immediately and irrevocably. In general, we associate a distinct time step with each good and it is often (although not always) convenient to implicitly rename the goods so that g_k is the k-th good in order of arrival and arrived at (or rather *triggered*) time step $t = k$. Also, we often use $n_h(i, t) = |\{g \in M \mid v_i(g) = \alpha_i \text{ and } g \text{ is one of the first } t \text{ goods that arrived}\}|$ to denote the number of high-valued goods among the first t goods from the perspective of agent i.

As we mentioned in our introduction, there are no distributional assumptions about the arrival of the goods and in our results we follow a worst-case analysis. In Section 5, however, we assume that our instances can be augmented with limited information about the future. We say that an online instance is augmented with *foresight of length* ℓ if every time a good g arrives (and still needs to be allocated immediately and irrevocably), we also get a preview of the ℓ next goods.

We mentioned above that we want to produce allocations that are *fair* in some manner. We formalize this by introducing our main fairness notions, *approximate EF*k and *approximate MMS*. Envy-freeness up to k goods (EFk) is a relaxation of envy-freeness, introduced by Lipton et al. [2004] and formally defined by Budish [2011] for $k = 1$. For instance, according to EF1 some envy is acceptable, as long as it can be eliminated by the hypothetical removal of a single good.

Definition 2.3 (ρ-Envy-Freeness, ρ-EFk)**.** Given a partial allocation $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, constants $\rho \in (0, 1]$ and $k \in \mathbb{Z}_{>0}$, and two agents $i, j \in N$, we say that

- agent i *is ρ-envy-free (ρ-EF) towards* agent j, if $v_i(A_i) \geq \rho \cdot v_i(A_j)$;
- agent i *is ρ-EFk towards* agent j, if there is a set $S \subseteq A_j$ with $|S| \leq k$, such that $v_i(A_i) \geq \rho \cdot v_i(A_j \setminus S)$.

The allocation is called ρ-EF (resp. α-EFk) if every agent $i \in N$ is ρ-envy-free (resp. ρ-EFk) towards any other agent $j \in N$. When $\rho = 1$, we drop the prefix and write EFk rather than 1-EFk.

Maximin share fairness, introduced by Budish [2011], is a share-based notion, like proportionality, and can be interpreted via a thought experiment inspired by the famous cut-and-choose protocol. The idea is to give to each agent at least as much value as it could get if it partitioned the goods into disjoint sets (as many as the agents) and kept the worst among them.

Definition 2.4 (ρ-PROP, ρ-MMS)**.** For a partial allocation $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, let $\Pi(n, \mathcal{A})$ be the set of possible partitions of the set $S = \bigcup_{j=1}^{n} A_j$ into n subsets. Given $\mathcal{A}$ above, a constant $\rho > 0$, and an agent $i \in N$, we say that

- $\mathcal{A}$ *is ρ-proportional* for agent i, if $v_i(A_i) \geq \rho \cdot v_i(S)/n$;

- $\mathcal{A}$ *is* ρ*-MMS* for agent i, if $v_i(A_i) \geq \rho \cdot \boldsymbol{\mu}_i^n(S)$, where $\boldsymbol{\mu}_i^n(S) = \max_{\mathcal{B} \in \Pi(n,\mathcal{A})} \min_{B_j \in \mathcal{B}} v_i(S)$ is the *maximin share* of agent i; when n is clear and S is a function of time t, we write $\boldsymbol{\mu}_i(t)$ instead $\boldsymbol{\mu}_i^n(S)$.

The allocation is called ρ-PROP (resp. ρ-MMS) if it is ρ-proportional (resp. ρ-MMS) for every agent $i \in N$. When $\rho = 1$, we write MMS rather than 1-MMS.

It is known, and very easy to derive from the definitions, that ρ-envy-freeness implies not only ρ-envy-freeness up to k goods, for any $k > 0$, but also ρ-proportionality, which itself implies ρ-maximin share fairness.

As our problem is online, we do not only care about the final (complete) allocation. As a result, we will make statements of the form "the allocation at time step t is ρ_1-EFk (resp. ρ_2-MMS)" meaning that we consider and evaluate the allocation that has been constructed up to time step t as if the complete set of goods was only what we have seen so far. If each one of the partial allocations produced by an algorithm satisfies the same fairness guarantee, then one talks about *temporal fairness*, as this was formalized by Elkind et al. [2024] and Cookson et al. [2025].

Definition 2.5 (Temporal Fairness)**.** Consider a sequence of partial allocations $\mathcal{A}^t = (A_1^t, A_2^t, \ldots, A_n^t)$, for $t \in \mathbb{Z}_{\geq 0}$, such that $A_i^t \subseteq A_i^{t+1}$ for any $i \in N$ and any $t \geq 0$. If $\mathcal{A}^t$ is ρ_1-EFk (resp. ρ_2-MMS) for all $t \in \mathbb{Z}_{\geq 0}$, then we say that the sequence of allocations $(\mathcal{A}^t)_{t \geq 0}$ is ρ_1-*temporal*-EFk (resp. ρ_2-*temporal*-MMS).

When referring to the allocation iteratively built by an algorithm, we may abuse the terminology and say that the algorithm computes a ρ_1-temporal-EFk (resp. ρ_2-temporal-MMS) allocation, rather than talking about a sequence of allocations.

Remark 2.6. Suppose that we have a personalized interval-restricted instance or a personalized 2-value instance with type 1 agents. Let $\alpha_* = \max_{i \in [n]} \sqrt{\alpha_i}$. It should be noted that here one can get a trivial $1/\alpha_*$ approximation with respect to temporal EF1 or MMS. Indeed, this is done by completely ignoring the values and allocating the goods in a round-robin fashion. Although, in general, $1/\alpha_*$ can be arbitrarily worse than the approximation factors we achieve throughout this work, in the special case where α_* is a small constant (e.g., between 1 and 2), it would be preferable to follow this trivial approach instead.

3 Impossibility Results Persist Even for 2-Value Instances

The strong impossibility results in the literature [He et al., 2019, Zhou et al., 2023] typically exploit the following pattern: a bad decision is made about the very first good, due to lack of information, and this propagates throughout a linear number of goods; then, right about when the value of the allocated goods starts to balance out, this "bad" sequence is replicated but with all values scaled up by a large factor, and this cycle is repeated as necessary. One could hope that for 2-value instances there is not enough flexibility for creating instances that force any algorithm to perform poorly. While there is some truth in this, in the sense that things cannot go arbitrarily bad, we still get nontrivial impossibility results.

Theorem 3.1. *Let $\varepsilon > 0$ be a constant. There is no deterministic algorithm that always builds an allocation which is $(1/2 + \varepsilon)$-temporal-EF1, even for 2-value instances with only two agents.*

Proof. Suppose we have a deterministic algorithm $\mathcal{A}$ for the problem. We begin with the simple observation that when the very first good has the same value for both agents, it is without loss of generality to assume that $\mathcal{A}$ assigns it to agent 1; if not, we just rename the agents in the following argument.

So, consider a stream of goods that begins with g_1, g_2, such that $v_1(g_1) = v_2(g_1) = 1$ whereas $v_1(g_2) = 5$ and $v_2(g_2) = 1$. Given that g_1 is added to A_1, either g_2 is also added to A_1 and the resulting allocation $(\{g_1, g_2\}, \emptyset)$ is not $(1/2 + \varepsilon)$-EF1 from the point of view of agent 2, or g_2 is added to A_2; we assume the

latter. Next, consider a good g_3, such that $v_1(g_3) = v_2(g_3) = 1$. There are two cases here, depending on what $\mathcal{A}$ does with g_3, both illustrated below:

	g_1	g_2	g_3	g_4	$\dots$		g_1	g_2	g_3	g_4	g_5	$\dots$
agent 1:	(1)	5	(1)	5	$\dots$	or	(1)	5	1	1	5	$\dots$
agent 2:	1	(1)	1	5	$\dots$		1	(1)	(1)	5	5	$\dots$

Case 1: g_3 is given to agent 1. In this case, consider a next good g_4, such that $v_1(g_4) = v_2(g_4) = 5$. Whoever gets g_4, the resulting allocation, $(\{g_1, g_3, g_4\}, \{g_2\})$ or $(\{g_1, g_3\}, \{g_2, g_4\})$, is not $(1/2 + \varepsilon)$-EF1. Indeed, $(\{g_1, g_3, g_4\}, \{g_2\})$ is not $(1/2 + \varepsilon)$-EF1 from the point of view of agent 2, since

$$1 = v_2(A_2) < (1/2 + \varepsilon) \min_{g \in A_1} (v_2(A_1) - v_2(g)) = 1 + 2\varepsilon\,.$$

Similarly, $(\{g_1, g_3\}, \{g_2, g_4\})$ is not EF1 from the point of view of agent 1, since now we have

$$2 = v_1(A_1) < (1/2 + \varepsilon) \min_{g \in A_2} (v_1(A_2) - v_1(g)) = 2.5 + 5\varepsilon\,.$$

Case 2: g_3 is given to agent 2. Here we first have an intermediate good g_4, such that $v_1(g_4) = 1$ and $v_2(g_4) = 5$. It is straightforward to see that if $\mathcal{A}$ assigns g_4 to agent 2, then the resulting allocation $(\{g_1\}, \{g_2, g_3, g_4\})$ is not $(1/2 + \varepsilon)$-EF1 from the point of view of agent 1. Hence, we assume that g_4 is added to A_1 instead. The last good we need is g_5 with $v_1(g_5) = v_2(g_5) = 5$. Now, no matter who gets g_5, the resulting allocation, $(\{g_1, g_4, g_5\}, \{g_2, g_3\})$ or $(\{g_1, g_4\}, \{g_2, g_3, g_5\})$, is not $(1/2 + \varepsilon)$-EF1. To see this, first consider $(\{g_1, g_4, g_5\}, \{g_2, g_3\})$ and notice that from the point of view of agent 2, we have

$$2 = v_2(A_2) < (1/2 + \varepsilon) \min_{g \in A_1} (v_2(A_1) - v_2(g)) = 3 + 6\varepsilon\,.$$

Finally, consider $(\{g_1, g_4\}, \{g_2, g_3, g_5\})$. From the point of view of agent 1, we have a similar situation:

$$2 = v_1(A_1) < (1/2 + \varepsilon) \min_{g \in A_2} (v_1(A_2) - v_1(g)) = 3 + 6\varepsilon\,.$$

In any case, algorithm $\mathcal{A}$ fails to maintain a $(1/2 + \varepsilon)$-EF1 allocation within the first 5 time steps. □

By carefully inspecting the proof of Theorem 3.1, one could notice that the same construction can be used to show that no algorithm can build an allocation that is $(1/3 + \varepsilon)$-MMS at every time step. In fact, the latter would imply Theorem 3.1 since it is known that any $(1/2 + \delta)$-EF1 allocation for two agents is also a $(1/3 + \delta/9)$-MMS allocation (see, e.g., Proposition 3.6 of Amanatidis et al. [2018]). Nevertheless, for temporal maximin share fairness we can show a much stronger impossibility result that degrades with the number of agents.

Theorem 3.2. *Let $\varepsilon > 0$ be a constant. There is no deterministic algorithm that, given a 2-value instance with n agents, always builds a $(1/(2n-1) + \varepsilon)$-temporal-MMS allocation.*

Proof. Suppose we have a deterministic algorithm $\mathcal{A}$ for the problem. We are going to consider a 2-value instance with n agents, such that, for all $i \in N$, $\beta_i = 1$ and $\alpha_i = \alpha = 2n^2 + 2n$. Like in the proof of Theorem 3.1, we begin with some straightforward, yet crucial observations. First, if at any point during the first n time steps an agent receives a second good, by the time the first n goods are fully allocated, there is at least one agent j that has received value 0 (because it got no goods) despite having a positive maximin share value $\boldsymbol{\mu}_j(n) \geq 1$. So, for what follows, we may assume that algorithm $\mathcal{A}$ assigns to each agent exactly one of the first n goods. A second observation is that, given that $\ell < n$ goods have been

already allocated, if the $(\ell+1)$-th good has the same value for all the agents who have not yet received a good, then it is without loss of generality to assume that $\mathcal{A}$ assigns it to the agent with the smallest index; this is just a matter of renaming the agents.

Given these two observations above, suppose that goods $g_1, \ldots, g_n$ arrive, in this order, so that for any $i \in N$

$$v_i(g_r) = \begin{cases} 1\,, & \text{if } i \leq r \\ \alpha\,, & \text{otherwise.} \end{cases}$$

Then, algorithm $\mathcal{A}$ assigns them exactly as shown below, i.e., g_i is given to agent i, for all $i \in N$.

	g_1	g_2	g_3	$\cdots$	g_{n-1}	g_n	g_{n+1}	g_{n+2}	$\cdots$	g_{2n-1}
ag. 1:	(1)	α	α	$\ldots$	α	α	1	1	$\ldots$	1
ag. 2:	1	(1)	α	$\ldots$	α	α	1	1	$\ldots$	1
ag. 3:	1	1	(1)	$\ldots$	α	α	1	1	$\ldots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
ag. $n-1$:	1	1	1	$\ldots$	(1)	α	1	1	$\ldots$	1
ag. n:	1	1	1	$\ldots$	1	(1)	1	1	$\ldots$	1

At this point it is not hard to see that the next claim about goods that are low-valued for all holds. Notice that $1/2n < 1/(2n-1) + \varepsilon$.

Claim 3.3. *Assume that the allocation of the first n goods is as shown above. If at any point $t > n$ no agent $i \leq \lambda$ has yet received total value more than $n+1$ and, furthermore, λ goods, $g'_1, \ldots, g'_\lambda$, which are high-valued for all agents arrive in that order, then either agent j will get good g'_j, for all $j \in [\lambda]$, or algorithm $\mathcal{A}$ builds at some point an allocation that is not $1/2n$-MMS.*

Proof of Claim 3.3. This is a simple proof by induction on λ. For $\lambda = 1$, after g'_1 appears, say at time t', we have $n_h(1, t') = n$, i.e., agent 1 has already seen n high-valued goods (from its own perspective). So, we have $\boldsymbol{\mu}_1(t') \geq \alpha$. On the other hand, if $\mathcal{A}$ does not add g'_1 to A_1, we have $v_1(A_1) \leq n+1 \leq (n+1)\boldsymbol{\mu}_1(t')/\alpha = \boldsymbol{\mu}_1(t')/2n$.

Next, for $\lambda > 1$, assume that the claim is true for $\lambda - 1$ agents and high-valued goods. Now suppose that no agent $i \leq \lambda$ has received value more than $n+1$ yet and that λ goods, $g'_1, \ldots, g'_\lambda$, which are high-valued for all agents arrive in that order. By this induction hypothesis, either algorithm $\mathcal{A}$ builds at some point an allocation that is not $1/2n$-MMS or, for all $j \in [\lambda - 1]$, agent j will get good g'_j. Given that, agent λ has now seen n high-valued goods in total and we can repeat the argument we made for agent 1 in the base case. Let t'' be the time step when g'_λ arrives. We have $n_h(\lambda, t'') = n$ and, thus, $\boldsymbol{\mu}_\lambda(t'') \geq \alpha$, whereas $v_\lambda(A_\lambda) \leq n+1 \leq \boldsymbol{\mu}_\lambda(t'')/2n$, unless $\mathcal{A}$ adds g'_λ to A_λ. This completes the induction step. CI. 3.3 ⊡

The next $n-1$ goods, $g_{n+1}, \ldots, g_{2n-1}$, are low-valued for all agents. Clearly, no matter how $\mathcal{A}$ assigns these goods, there is at least one agent, say $k \in N$, who does not get any of those by the end of time step $2n-1$. We will distinguish two cases, depending on whether agent k is the last agent or not.

Case 1: $k = n$. In this case, the next $n-1$ goods, $g_{2n}, \ldots, g_{3n-2}$, are high-valued for all agents. Given that no agent has received total value more than n by the end of time step $2n-1$, Claim 3.3 applies, forcing the algorithm $\mathcal{A}$ to either fail or allocate $g_{2n}, g_{2n+1}, \ldots, g_{3n-2}$ to agents 1, 2, …, $n-1$, in that order. We assume the latter. After this happens we notice that $\boldsymbol{\mu}_n(3n-2) = 2n-1$, since agent n has already seen $n-1$ high-valued and $2n-1$ low-valued goods, and $\alpha > (2n-1) \cdot 1$. On the other hand, $A_n = \{g_n\}$, i.e., we have $v_n(A_n) = 1 = \boldsymbol{\mu}_n(3n-2)/(2n-1) < (1/(2n-1) + \varepsilon)\boldsymbol{\mu}_n(3n-2)$.

Case 2: $k < n$. In this case, the next $n-k$ goods, $g_{2n}, \ldots, g_{3n-k-1}$, are such that, for $\ell \in [n-k]$ and for $i \in N$,

$$v_i(2n-1+\ell) = \begin{cases} \alpha\,, & \text{if } i = n-\ell+1 \\ 1\,, & \text{otherwise} \end{cases}$$

i.e., g_{2n} is high-valued only for agent n, g_{2n+1} only for agent $n-1$, and so on, as is shown in the left part of the table below. Note, however, that we may not see the whole subsequence $g_{2n}, \ldots, g_{3n-k-1}$, depending on the behavior of algorithm $\mathcal{A}$.

	g_{2n}	g_{2n+1}	$\cdots$	g_{3n-k-1}	g_{3n-k}	g_{3n-k+1}	$\cdots$	g_{3n-2}
ag. 1:	1	1	…	1	$\circled{\alpha}$	α	…	α
ag. 2:	1	1	…	1	α	$\circled{\alpha}$	…	α
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
ag. $k-1$:	1	1	…	1	α	α	…	$\circled{\alpha}$
ag. k:	1	1	…	1	α	α	…	α
ag. $k+1$:	1	1	…	$\circled{\alpha}$	α	α	…	α
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
ag. $n-1$:	1	$\circled{\alpha}$	…	1	α	α	…	α
ag. n:	$\circled{\alpha}$	1	…	1	α	α	…	α

We claim that either algorithm $\mathcal{A}$ fails to maintain a $(1/(2n-1)+\varepsilon)$-MMS allocation or it allocates $g_{2n}, g_{2n+1}, \ldots, g_{3n-k-1}$ to agents n, $n-1$, …, $k+1$, in that order. Towards a contradiction, suppose that this is not the case, i.e., $\mathcal{A}$ maintains a good approximation to temporal maximin share fairness, yet some of these goods are not allocated to the agent who values them highly. In particular, let $2n-1+j \in \{2n, 2n+1, \ldots, 3n-k-1\}$ be the lowest index of a good for which this happens. That is, g_{2n} is given to agent n, g_{2n+1} to agent $n-1$, and so on, g_{2n-2+j} is given to agent $n-j+2$, *but* g_{2n-1+j} is not given to agent $n-j+1$. As a result, no agent $i \in [n-j+1]$ has received total value more than $n+1$ by the end of time step $2n-1+j$. So, at this point we may change the stream of goods, forget about $g_{2n+j}, \ldots, g_{3n-k-1}$ above, and replace them by $n-j$ goods, $\hat{g}_{2n+j}, \ldots, \hat{g}_{3n-1}$, which are high-valued for all agents. As we argued already, Claim 3.3 applies here, forcing $\mathcal{A}$ to either fail or allocate $\hat{g}_{2n+j}, \hat{g}_{2n+j+1}, \ldots, \hat{g}_{3n-1}$ to agents 1, 2, …, $n-j$, respectively, in that order. We assume the latter. After this happens we notice that $\boldsymbol{\mu}_{2n-1+j}(3n-1) \geq \alpha$, since agent $2n-1+j$ has already seen n high-valued goods ($j-1$ among the first n goods, the good g_{2n-1+j} itself, and all of the last $n-j$ goods). On the other hand, A_{2n-1+j} only contains at most $n+1$ low-valued goods. That is, we have $v_{2n-1+j}(A_{2n-1+j}) \leq n+1 \leq (n+1)\boldsymbol{\mu}_{2n-1+j}(3n-1)/\alpha < \boldsymbol{\mu}_n(3n-1)/2n$.

At this point we may assume that algorithm $\mathcal{A}$ allocates $g_{2n}, g_{2n+1}, \ldots, g_{3n-k-1}$ to agents n, $n-1$, …, $k+1$, in that order, as shown in the corresponding table. However, this means that none of the first k agents has received more than n low-valued goods so far. In particular, $A_k = \{g_k\}$. As a result we can consider $k-1$ additional goods, $g_{3n-k}, \ldots, g_{3n-2}$, which are high-valued for all agents and apply Claim 3.3. The claim guarantees that either algorithm $\mathcal{A}$ does not maintain a sufficiently high approximation guarantee throughout, or that the allocation will be completed as shown in the last part of the second table, i.e., $g_{3n-k}, g_{3n-k+1}, \ldots, g_{3n-2}$ are given to agents 1, 2, …, $k-1$, respectively, in that order. This is a very poor allocation from the point of view of agent k. We notice that $\boldsymbol{\mu}_k(3n-2) = 2n-1$, since agent k has already seen $n-1$ high-valued goods ($n-k$ among the first n goods and all of the last $k-1$ goods) and $2n-1$ low-valued goods (k among the first n goods and all the $(n-1)+(n-k)$ goods right after the first n goods). On the other hand, recall that $A_k = \{g_k\}$. Thus, we have $v_k(A_k) = 1 = \boldsymbol{\mu}_n(3n-2)/(2n-1) < (1/(2n-1)+\varepsilon)\boldsymbol{\mu}_n(3n-2)$. □

By inspecting the proof, it is not hard to see that Theorem 3.2 could have been stated with respect to the largest ratio of the high over the low value of an agent. This is particularly relevant for Section 6, where this term will appear in the fairness guarantees we obtain for personalized interval-restricted instances.

Corollary 3.4. *Let $\varepsilon > 0$ be a constant. There is no algorithm that, given a 2-value instance with n agents and values $\alpha > 1 = \beta$, always builds a $(1/\sqrt{2\alpha} + \varepsilon)$-temporal-MMS allocation.*

Proof. Notice that in the proof of Theorem 3.2 we have $\alpha = 2n^2 + 2n$. Also, by standard calculus we get $\lim_{n\to\infty} \left(\frac{1}{2n-1} - \frac{1}{\sqrt{4n^2+4n}}\right) = 0$. That is, for large enough n, it holds that $\frac{1}{\sqrt{4n^2+4n}} + \varepsilon > \frac{1}{2n-1} + \frac{\varepsilon}{2}$ and the impossibility follows directly by Theorem 3.2. □

4 A Tight Algorithm

In this section we present the main result of this work, an algorithm with tight temporal maximin share fairness guarantees. Given how nuanced the construction of the example in the proof of Theorem 3.2 was, it is not particularly surprising that matching this $1/(2n-1)$ factor requires a fairly elaborate algorithm that performs careful book-keeping of who should get the next contested high-valued goods.

Before discussing any details of our Deferred-Priority algorithm (Algorithm 1), we revisit the observation that we made at the beginning of the proof of Theorem 3.2. When agents have positive values, if we aim for a nonzero temporal maximin share fairness guarantee, Algorithm 1 should assign to each agent exactly one of the first n goods. Indeed, if this is not the case, there would be at least one agent j who receives value 0 (because j got no goods) at the end of time step n, despite having a positive maximin share value $\boldsymbol{\mu}_j(n) \geq \beta_i = 1$. So, no matter how our algorithm works in general, the allocation of the first n goods is "special" in the sense that it allows for extremely little flexibility. We call these first n time steps Phase 0. More generally, Algorithm 1 will operate in phases; we want to ensure that during each phase every agent gets at least one good and the phases do not last for too long.

As a second general design goal, however, we want to allocate goods to agents who consider them high-valued as frequently and as uniformly as possible. Note that this is rather incompatible with the aforementioned goal of frequently giving goods to everyone. Our solution to that is to have two sets of counters (the entries of the vectors H and L, introduced in line 3) that keep track of how many high- or low-valued goods, respectively, an agent affords to lose to others before we are in a situation where a "bad" sequence of goods inevitably destroys the temporal maximin share fairness guarantee.

So, the general idea is that throughout the allocation the agents should gain higher priority the more value they lose to others; the corresponding *priority levels* are implicitly described by the entries of H and L (and later on explicitly defined for high-valued goods as $\mathcal{H}_\ell(t)$). Indeed, every time a good arrives and is allocated, the entries of H and L are updated accordingly. The way these indices are updated enforces that one out of every $2n-1$ goods, in general, and one out of n—that later becomes $3n-2$—high-valued goods is allocated to each agent. These quantities may seem somewhat loose, but as it shown in our analysis, asking for more frequent allocations per agent, would not leave enough room for our competing goals to work simultaneously.

In our proofs and statements we often need to refer to the agents' bundles at different time steps. For clarity, we write A_i^t (rather than just A_i) to denote the bundle of agent i at the end of time step t. In fact, we use this notation to the statement of the main theorem of this section as well. Recall also that $n_h(i,t) = |\{g \in M \mid v_i(g) = \alpha_i \text{ and } g \text{ is one of the first } t \text{ goods that arrived}\}|$ is the number of high-valued goods agent i has seen up to (and incuding) time t.

Algorithm 1 Deferred-Priority$(v_1, \ldots, v_n; M)$

(The valuation functions, v_i, $i \in [n]$, are given via oracles; M is given in an online fashion, one good at a time.)

$\texttt{phase} \leftarrow 0$; $\texttt{low} \leftarrow 0$; $\texttt{high} \leftarrow 0$; $t \leftarrow 0$ // We initialize all of our counters.
for $i \in N$ **do**
    $A_i \leftarrow \emptyset$; $H[i] \leftarrow n$; $L[i] \leftarrow 2n-1$; $\chi_i \leftarrow 0$ // Ag. i tolerates the loss of less than n high-valued goods.
whenever a new good g **arrives:**
$t \leftarrow t+1$
for $i \in N$ **do**
    if $v_i(g) = \alpha_i > 0$ **then**
        $H[i] \leftarrow H[i]-1$ // Potential loss of a high-valued good; i's priority for high-valued goods is increased.
    else
        $L[i] \leftarrow L[i]-1$ // Potential loss of a low-valued good; i's priority for low-valued goods is increased.
$N_h(g,t) \leftarrow \{i \in N \mid v_i(g) = \alpha_i \text{ and } \chi_i = 0\}$ // Potential recipients of g as a high-valued good.
$N_\ell(g,t) \leftarrow \{i \in N \mid v_i(g) = \beta_i \text{ and } \chi_i = 0\}$ // Potential recipients of g as a low-valued good.
if $N_h(g,t) \neq \emptyset$ **then**
    $\texttt{high} \leftarrow \texttt{high}+1$ // g is allocated as a high-valued good.
    $j = \arg\min_{i \in N_h(g,t)} H[i]$ // Ag. j has the highest priority for g; ties are broken lexicographically.
    $A_j \leftarrow A_j \cup \{g\}$ // The good is added to ag. j's bundle.
    $H[j] \leftarrow H[j]+3n-2$ // Now ag. j tolerates the loss of at most $3n-2$ high-valued goods.
    $\chi_j \leftarrow 1$ // Ag. j will not get any more goods during the current phase.
else
    $\texttt{low} \leftarrow \texttt{low}+1$ // g is allocated as a low-valued good.
    $j = \arg\min_{i \in N_\ell(g,t)} L[i]$ // Ag. j has the highest priority for g; ties are broken lexicographically.
    $A_j \leftarrow A_j \cup \{g\}$ // The good is added to ag. j's bundle.
    $L[j] \leftarrow 2n+t$ // Ag. j now has the lowest priority among active agents for the rest of the phase.
    if $\texttt{phase} = 0$ **then**
        $\chi_i \leftarrow 1$ // If this is phase 0, ag. j will not get any more goods.
if ($\texttt{phase} = 0$ **and** $\texttt{low} + \texttt{high} = n$) **or** ($\texttt{phase} > 0$ **and** $\max\{\texttt{low}, \texttt{high}\} = n$) **then**
    $\texttt{phase} \leftarrow \texttt{phase}+1$ // The conditions to conclude this phase were met and we move to the next one.
    $\texttt{low} \leftarrow 0$; $\texttt{high} \leftarrow 0$ // We reset our counters.
    for $i \in N$ **do**
        $L[i] \leftarrow 2n-1$ // We reset the priority for low-valued goods.

Theorem 4.1. *Algorithm 1 builds an allocation such that, for every $i \in N$:*

1. $|A_i^n| = 1$, *i.e., agent i gets one of the first n goods (assuming $m \geq n$; otherwise $|A_i^m| \leq 1$).*
2. *At (the end of) any time step t, A_i^t contains at least $\lfloor n_h(i,t)/(3n-2) \rfloor$ high-valued goods. Moreover, if agent i gets to see at least n high-valued goods, at (the end of) time step $t_{i0} = \min\{t \mid n_h(i,t) \geq n\}$, $A_i^{t_{i0}}$ contains at least 1 high-valued good.*
3. *At (the end of) any time step $t \geq n$, $|A_i^t| \geq \lfloor (t-n)/(2n-1) \rfloor + 1$, i.e., agent i has received at least one out of every $2n-1$ goods they have seen after the first n goods.*

Proof of parts 1. and 3. We are going to show parts 1., 2., and 3. separately. While parts 1. and 3. are relatively straightforward, part 2. requires an elaborate analysis using delicate inductive arguments. During any phase, an agent i is called *active* if $\chi_i = 0$ and *inactive* otherwise. As it is clear by the definition of the sets

$N_h(g,t)$ and $N_\ell(g,t)$ (lines 11 and 12), no agent can receive any more goods during the current phase once it becomes inactive.

We begin with part 1. of the theorem. Lines 18 and 24-25 update χ_i to 1 for any agent who receives *any* good during Phase 0, ensuring that no one gets more than 1 good during this phase. Further, if $m \geq n$, lines 14 and 20, combined with the *first* part of the condition in line 26, ensure that exactly n goods are allocated during Phase 0, as either low-valued or high-valued goods. Therefore, $|A_i^n| = 1$, i.e., each agent gets exactly 1 of the first n goods.

Moving to part 3., let $q \in \mathbb{Z}^+$. We observe that, during Phase q, no more than $2n-1$ goods are allocated, as it is enforced by lines 14 and 20, combined with the *second* part of the condition in line 26. Next, note that, like in Phase 0, if an agent i receives a high-valued good (which triggers χ_i to become 1 in line 18), becomes inactive and never receives another good during Phase q. However, unlike in Phase 0, when agent i receives a low-valued good at time t, it now stays active. Nevertheless, in such a case, agent i becomes *last* in the implicit priority list of active agents for low-valued goods, as now $L[i] = 2n + t$ (line 23) and $L[j] \leq 2n + t'$ with $t' < t$, for $j \in N \setminus \{i\}$. The important observation here is that once this happens, agent i cannot receive another good before every other agent is inactive or receives a low-valued good at a time $t'' > t$. We are now ready for the main argument that implies part 3.

If Phase q terminates because $\texttt{high} = n$, then every agent received exactly one high-valued good. Otherwise, if Phase q terminates because $\texttt{low} = n$, we distinguish two simple cases. If no agent got more than one low-valued good in Phase q, then everyone received exactly one low-valued good (and at least one good, in general). So, assume that there is some agent i who received at least 2 low-valued goods during Phase q. By the preceding discussion, this means that every other agent became inactive at some point (by receiving a high-valued good) or received a low-valued good between the times when agent i got its first and second low-valued good. In any case, if Phase q terminates, then each agent has received at least 1 out of at most $2n-1$ goods that were allocated during the phase. Combining this with the fact that every agent gets 1 out of n goods in Phase 0 (by part 1.), we conclude that at the end of any time step t, $|A_i^t| \geq 1 + \lfloor (t-n)/(2n-1) \rfloor$. □

Most of the remaining section is dedicated to proving part 2. of Theorem 4.1. Thus, it would be useful to give some intuition behind both the statement and its proof. In doing so, we will establish some additional notation and terminology. The obvious way to show that each agent gets at least 1 out of the first n high-valued goods they see and 1 out of every $3n-2$ high-valued goods overall, is to show that it is always possible to allocate the goods in such a way so that $H[i] > 0$, for all $i \in N$, at the end of each time step t (i.e., right before the $(t+1)$-th good arrives). The reason, of course, is that $H[i]$ has been defined to contain the number of high-valued goods that agent i can afford to lose before the desideratum of part 2. of Theorem 4.1 is violated.

A straightforward necessary condition for $H[i] > 0, i \in N$, to hold at the end of each time step t is to have at most one agent j, such that $H[j] = 1$ before a new good arrives. To see this, assume that there are distinct j, j' such that $H[j] = H[j'] = 1$ and that the next good g that arrives is high-valued for everyone. Then, no matter how g is allocated, at least one of $H[j], H[j']$ will hit 0, meaning that enough high-valued goods were lost for one of the two agents for part 2. of Theorem 4.1 to fail. In fact, we can extend this necessary condition to having at most k agents $j_1, \ldots, j_k$, such that $H[j_\ell] \leq k$ before a new good arrives, for any $k \in [n]$. Indeed, if there are at least $k+1$ such agents and the next k goods are high-valued for everyone, it is impossible to allocate them without making some coordinate(s) of vector H equal to 0.

The interesting thing here, is that this simple necessary condition always allows us to "legally" allocate at least one next good g. Roughly speaking, if we consider the agents who see g as high-valued and give it to the agent with the smallest H entry among them, then it is not very hard to see (we will prove it formally

in Claim 4.3) that it is not possible to end up with any agent i having $H[i] = 0$ (recall that after receiving a good, an agent's H entry increases significantly). So, if one could allocate each good and, at the same time, maintain the condition that H contains at most k entries that are k or below, for all $k \in [n]$, then part 2. of Theorem 4.1 would follow. The tricky part is to make sure that this condition still holds after allocating *any* good and this is the core of the technical difficulty of proving the theorem, mainly because we often need to allocate goods that are not low-valued for everyone to agents who see them as low-valued. In fact, the latter is absolutely necessary for part 3. of the theorem shown above.

In order to formalize things, we introduce the following notation for the *level sets* that contain all agents with the same priority according to H at any given time:

$$\mathcal{H}_\ell(t) = \{i \in N \mid H[i] = \ell \text{ at the end of time step } t\}\,.$$

With this notation, the above necessary and sufficient condition for being able to legally extend the allocation at time $t+1$ becomes

$$\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k, \text{ for all } 0 \leq k \leq n\,. \tag{1}$$

In the discussion above, we imply that maintaining (1) is easier if, whenever a good is viewed as high-valued by someone, it is always allocated as a high-valued good. Indeed, this is the case: if we only allocated goods so as to maximize the social welfare, then we would be able to maintain (1) for every t, even if in line 17 we only added n rather than $3n-2$. The technical reason why will become clear in the proofs of Claims 4.6 and 4.7, but the issue with this is that it would mean that agents who mostly see low-valued goods might have to wait arbitrarily long before getting anything. Hence, we add $3n-2$ in line 17, to give us some extra room to keep every agent content. From a technical point of view, within our proofs we typically need to decouple the two cases that cause changes to entries of H (allocating a good that is not globally low-valued as high-valued *versus* as low-valued), as they are qualitatively very different.

The following lemma states the fact that condition (1) holds for all time steps $t \geq 0$. As its proof is fairly long and complicated, it is deferred to the Section 4.1.

Lemma 4.2. *At the end of any time step $t \geq 0$, we have $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k$, for all $0 \leq k \leq n$.*

At this point we are ready to prove part 2. of Theorem 4.1.

Proof of part 2. of Theorem 4.1. In the discussion preceding this proof, we claimed that condition (1) is sufficient, and we briefly argued about it in a rather hand-wavy way. Here we begin by formalizing this fact. It should be noted that Claim 4.3 does not say that by allocating the $(t+1)$-th good g we ensure that the condition holds for time step $t+1$; this is shown separately in Lemma 4.2. The claim barely states that whenever condition (1) holds and things have gone well in the past, Algorithm 1 can allocate the next good without violating the guarantees we aim to show.

Claim 4.3. *Let t be any time step in $\{0, 1, \ldots, m-1\}$ and let g be the $(t+1)$-th good. Assuming that all entries of H have remained positive at the end of all time steps up to t, condition (1) guarantees that Algorithm 1 can allocate g without any entry of H becoming 0 at the end of time step $t+1$.*

Proof of Claim 4.3. For the sake of clarity, here we make the dependency of H on t explicit and write $H_t[i]$ to denote $H[i]$ at the end of time step t. Assume that $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k$, for all $k \in [n]$. In particular, $|\mathcal{H}_1(t)| \leq 1$. If $|\mathcal{H}_1(t)| = 0$, then $H_t[i] \geq 2$ for all $i \in N$ and clearly, no matter how good g is allocated, $H_{t+1}[i] \geq 1$ for all $i \in N$. Similarly, if $|\mathcal{H}_1(t)| = 1$ and $j \in N$ is the unique agent such that $H_t[j] = 1$

but $v_j(g) = \beta_j$, we have $H_{t+1}[j] = H_t[j] = 1$ as well as $H_{t+1}[i] \geq H_t[i] - 1 \geq 1$ for all $i \in N \setminus \{j\}$, like before, no matter how g is allocated.

The interesting case here is when $|\mathcal{H}_1(t)| = 1$, $j \in N$ is the unique agent such that $H_t[j] = 1$ and $v_j(g) = \alpha_j$. Again, for $i \in N \setminus \{j\}$, $H_{t+1}[i] \geq 1$ but now we must show that Algorithm 1 will add g to A_j. Given that j has the highest priority (i.e., lowest entry in H, which has temporarily dropped to 0), for j to get g it suffices to show that $\chi_j = 0$. Towards a contradiction, assume this is not the case, i.e., j has received a high-valued good at time t' which belongs to the same phase as t. Since each phase has at most $2n - 1$ time steps (see the proof of part 3. of Theorem 4.1), $t - t' \leq 2n - 2$. But then,

$$H_t[j] \geq H_{t'}[j] - (2n - 2) = H_{t'-1}[j] - 1 + 3n - 2 - 2n + 2 \geq n\,,$$

contradicting the choice of j . We conclude that an agent j with $H_t[j] = 1$ cannot have received a high-valued good in the phase that includes t, thus $\chi_j = 0$. Therefore, j is the agent who gets g in line 15, and $H_{t+1}[j] = H_t[j] - 1 + 3n - 2 = 3n - 2 > 0$, completing the proof. Cl. 4.3 □

Recall that, by the design of the priority vector H, in order to show that each agent gets at least 1 out of the first n high-valued goods they see and 1 out of every $3n - 2$ high-valued goods overall, it suffices to show that it is always possible to allocate the goods so that $H[i] > 0$, for all $i \in N$, at the end of each time step t. By combining Claim 4.3 with Lemma 4.2, which shows that condition (1) is maintained throughout the execution of Algorithm 1, we have exactly that. The algorithm allocates all goods without any entry of H becoming 0 at the end of any time step. Equivalently, by the definition of how H is updated, agent i receives at least 1 high-valued good by time $t_{i0} = \min\{t \,|\, n_h(i,t) \geq n\}$ and at least 1 out of every $3n - 2$ high-valued goods they see after that, for a total of at least $\lfloor n_h(i,\tau)/(3n-2) \rfloor$ high-valued goods by the end of a time step $\tau \geq 0$. □

Now, using Theorem 4.1, we can argue about the temporal maximin share guarantees of the Deferred-Priority algorithm (Algorithm 1). Recall from Section 2 that an agent i is of type 1 when $\alpha_i > \beta_i = 1$, it is of type 2 when $\alpha_i = \beta_i = 1$ and it is of type 3 when $1 = \alpha_i > \beta_i = 0$. As the arguments needed from different types are somewhat different, we will state the corresponding guarantees separately.

Corollary 4.4. *Any agent i of type $k \in \{2, 3\}$ receives at least a constant fraction of its temporal maximin share by Algorithm 1. In particular, $v_i(A_i^t) \geq \boldsymbol{\mu}_i(t)/k$, for any time step $t \geq 0$.*

Proof. First let agent i be of type 2. We will bound the value i gets during the allocation sequence induced by Algorithm 1. At the end of any time step $t < n$, we have $\boldsymbol{\mu}_i(t) = 0$, so the statement trivially holds. If, instead, $n \leq t < n + (2n - 1)$, then by part 1. of Theorem 4.1 we have $|A_i^t| \geq 1$ and so, $v_i(A_i^t) \geq 1$, whereas $\boldsymbol{\mu}_i(t) \leq \lfloor (3n-1)/n \rfloor = 2$. Finally, we may assume that $n + k(2n-1) \leq t < n + (k+1)(2n-1)$, for some $k \in \mathbb{Z}_{>0}$. Then, by part 3. of Theorem 4.1, agent i has received at least one out of every $2n - 1$ goods they have seen after Phase 0 and so, $v_i(A_i^t) = |A_i^t| \geq \lfloor (t-n)/(2n-1) \rfloor + 1 = k + 1$, whereas, by the definition of maximin share,

$$\boldsymbol{\mu}_i(t) = \left\lfloor \frac{t}{n} \right\rfloor < \left\lfloor \frac{n + (k+1)(2n-1)}{n} \right\rfloor = \left\lfloor \frac{2(k+1)n + (n-k-1)}{n} \right\rfloor \leq 2(k+1)\,.$$

Next, assume that agent i is of type 3. Given that low-valued goods are completely irrelevant to agent i, we can consider a time alternative τ that starts at 0, like t, but only increases when a high-valued good for i arrives. That is, while t reflects how many goods have arrived in general, τ reflects how many high-valued goods with respect to agent i have arrived instead. We can repeat the exact same analysis we did for agents of type 2, but using τ instead of t, 3 instead of 2 for the factor, $3n - 2$ whenever the quantity $2n - 1$ was used, and by invoking part 2. of Theorem 4.1 instead of part 3. □

Corollary 4.5. *Any agent i of type 1 receives at least a $1/(2n-1)$ fraction of its temporal maximin share by Algorithm 1., i.e., $v_i(A_i^t) \geq \boldsymbol{\mu}_i(t)/(2n-1)$, for any time step $t \geq 0$, which improves to $\Omega(1)$ from time t_{i0} onward (recall that $t_{i0} = \min\{t \mid n_h(i,t) \geq n\}$).*

Proof. Let i be a type 1 agent and consider any time step t. Let κ, λ be the number of high-valued and low-valued goods in A_i^t, respectively, where $\kappa, \lambda \in \mathbb{Z}_{\geq 0}$.

Case 1: $\kappa = 0$. At the end of any time step $t < n$, we have $\boldsymbol{\mu}_i(t) = 0$ and the statement trivially holds. So assume that $t \geq n$ As $\kappa = 0$ implies that $n_h(i,t) \leq n-1$, or equivalently, $t < t_{i0}$ (by part 2. of Theorem 4.1), we have $n \leq t < t_{i0}$. On one hand, we know that $v_i(A_i^t) = \lambda \geq 1$, where the second inequality follows by part 1. of Theorem 4.1. On the other hand, by considering a hypothetical allocation where each one of the $n_h(i,t)$ high-valued goods for agent i is a whole bundle and the low-valued goods for agent i are split as equally as possible into $n - n_h(i,t)$ bundles, we see that i's maximin share is at most the value of the worst bundle among the ones filled with low-valued goods, i.e.,

$$\boldsymbol{\mu}_i(t) \leq \left\lfloor \frac{t - n_h(i,t)}{n - n_h(i,t)} \right\rfloor \leq \left\lfloor \frac{t - n_h(i,t) - (n - 1 - n_h(i,t))}{n - n_h(i,t) - (n - 1 - n_h(i,t))} \right\rfloor \leq \left\lfloor \frac{t - (n-1)}{1} \right\rfloor = t - (n-1)\,,$$

where the second inequality follows from $n_h(i,t) \leq n-1$ and the simple fact that $\frac{a}{b} \leq \frac{a-c}{b-c}$ for any $a \geq b > c \geq 0$. Note however that there is a straightforward upper bound on t, implied by parts 1. and 3. of Theorem 4.1: $t \leq n + (\lambda - 1)(2n-1) + (2n-2)$. Thus,

$$\boldsymbol{\mu}_i(t) \leq n + (\lambda-1)(2n-1) + (2n-2) - (n-1) = \lambda(2n-1) = (2n-1)v_i(A_i^t)\,.$$

For the remaining two cases, we are going to show an approximation factor of at least $1/4$ with respect to proportionality, which then implies the same guarantee for maximin share fairness. Let S be the set containing the first t goods for some t. Note that necessarily one of these two cases holds if $t \geq t_{i0}$ (but possibly even earlier than that).

Case 2: $\kappa \geqslant 1$ and $\lambda = 0$. In this easier case we have $v_i(A_i^t) = \kappa\alpha_i$ and i might have seen at most $\kappa(2n-1) + (2n-2)$ goods (by part 3. of Theorem 4.1) of total value $v_i(S) \leq (\kappa(2n-1) + (2n-2))\alpha_i$. We have

$$\boldsymbol{\mu}_i(t) \leq \frac{v_i(S)}{n} \leq \frac{\kappa(2n-1) + (2n-2)}{n}\alpha_i \leq \frac{2n\kappa_i + 2n}{n}\alpha_i = 2(\kappa+1)\alpha_i \leq 4\kappa\alpha_i = 4v_i(A_i^t)\,.$$

Case 3: $\kappa \geqslant 1$ and $\lambda \geqslant 1$. Similarly to Case 2, $v_i(A_i^t) = \kappa\alpha_i + \lambda$ and i might have seen at most $n + (\kappa + \lambda - 1)(2n-1) + (2n-2)$ goods (by parts 1. and 3. of Theorem 4.1), out of which at most $n + (\kappa-1)(3n-2) + (3n-3)$ can be high-valued for i (by parts 1. and 2. of Theorem 4.1). Then it is a matter of simple calculations to show that for the total value $v_i(S)$ to be maximized, the low-valued goods are at most $(\lambda-1)(2n-1) + (2n-2)$. That is,

$$\begin{aligned} v_i(S) &\leq [n + (\kappa-1)(3n-2) + (3n-3)]\alpha_i + [(\lambda-1)(2n-1) + (2n-2)] \\ &\leq [n + 3n(\kappa-1) + 3n]\alpha_i + 2n(\lambda-1) + 2n = 3n\Big(\kappa + \frac{1}{3}\Big)\alpha_i + 2n\lambda\,. \end{aligned}$$

Therefore, we have $\boldsymbol{\mu}_i(t) \leq v_i(S)/n \leq (3\kappa+1)\alpha_i + 2\lambda \leq 4\kappa\alpha_i + 4\lambda = 4v_i(A_i^t)$. □

4.1 Proving Lemma 4.2

Proof of Lemma 4.2. As we did in the proof of Claim 4.3, for clarity, we write $H_t[i]$ to denote $H[i]$ at the end of time step t. The proof will be broken down into two proofs, one for $t \leq n$ and one for $t \geq n$; for the sake of presentation, the corresponding cases of the lemma are stated as Claims 4.6 and 4.7 below.

Claim 4.6. *At the end of any time step $t \leq n$, we have $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k$, for all $0 \leq k \leq n$.*

Proof of Claim 4.6. We will use induction on the number of agents n for a slightly more general version of the algorithm that takes the initialization of $H = H_0$ as part of the input, where it must be that $H_0[i] \geq n$ for all $i \in N$. Then the statement of the claim follows by fixing this part of the input to be $H_0[i] = n$ for all $i \in N$.

For a single agent, it is straightforward that, for $H_0[1] \geq 1$, initially $\left|\mathcal{H}_0(0)\right| = 0$ and $\left|\mathcal{H}_1(0)\right| \leq 1$, whereas after the first good is allocated, either $H_1[1]$ remains unchanged and, thus, at least 1 (if the good was low-valued) or it is updated to $H_1[1] - 1 + 3 \cdot 1 - 2 \geq 1$ (if the good was high-valued). Either way, $\left|\mathcal{H}_0(1)\right| = 0$ and $\left|\mathcal{H}_1(1)\right| \leq 1$, completing our base case.

Now assume that the statement of the claim is true for a certain $n' \geq 1$, and consider any instance with $n = n'+1$ agents and any $H_0 \in \mathbb{Z}^n_{\geq n}$. For this particular instance, let $g_1, g_2, \ldots, g_m$ be the goods that arrive, in this order. Because of how goods are allocated in Phase 0, i.e., no agent gets a second good, once an agent j receives g_1 at time $t = 1$, the remaining of Phase 0 is indistinguishable from (the complete) Phase 0 of an instance with the agents of $N \setminus \{j\}$, an appropriate initial priority vector (defined by $H_0[i] - \mathbf{1}_{N_h(g_1,1)}(i)$ or $H_0[i] - \mathbf{1}_{N_h(g_1,1)}(i) - 1$; see Cases 1 and 2 below), and the sequence of goods being $g_2, g_3, \ldots, g_m$. We are going to invoke the induction hypothesis on this sub-instance but we distinguish two cases, depending on whether g_1 is allocated as a high-valued or a low-valued good. Notice that it is without loss of generality to assume that the agent who gets good g_1 is agent 1, as it is a matter of renaming the agents, if needed, and is consistent with our lexicographic tie-breaking.

For the sub-instance that only involves agents 2 through n, has a properly defined initial priority vector H'_0 (see within the cases for the corresponding description), and the sequence of goods is $g_2, \ldots, g_m$, we use a prime to distinguish the corresponding quantities. That is, we use t' to denote time, rather than t that we reserve for the original instance; in general $t' = t - 1$, e.g., the 5th time step in the sub-instance corresponds to the 6th time step of the original problem. Thus, we will write $H'_{t'}$ for the priority vector of the sub-instance at the end of time step t' *of that instance* and $\mathcal{H}'_\ell(t')$ for the level sets that it induces. Assuming that H'_0 is such that $H'_0[i] \geq n - 1$ for $i \in \{2, \ldots, n\}$, by the induction hypothesis, we have for this sub-instance:

$$\text{At the end of any } t' \leq n-1, \left|\bigcup_{\ell=0}^{k} \mathcal{H}'_\ell(t')\right| \leq k, \text{ for all } 0 \leq k \leq n-1\,. \tag{2}$$

Case 1: $v_1(g_1) = \alpha_1$. In this case, the initial vector H'_0 given as input for the sub-instance is defined by $H'_0[i] = H_0[i] - \mathbf{1}_{N_h(g_1,1)}(i) \geq n - 1$ for $i \in \{2, \ldots, n\}$, where $\mathbf{1}_S(i)$ is the indicator function of whether $i \in S$. Notice that this way, the priority among agents remains exactly the same as in the original instance and $H'_{t'}[i] = H_{t'+1}[i]$ for all $t' \in \{0, \ldots, n-1\}$ and $i \in \{2, \ldots, n\}$; of course, $H'_{t'}[1]$ is not defined. Further, because agent 1 gets a high-valued good at time 1, $H_1[1] = H_0[1]-1+3n-2 \geq 4n-3$ and, thus, throughout Phase 0 of the original instance (i.e., $t \in \{0, 1, \ldots, n\}$) we have $H_t[1] \geq 4n-3-(t-1) \geq 3n-2 \geq n$. So, agent 1 may only appear in $\mathcal{H}_n(t)$ for any $t \in [n]$. By the discussion about the correspondence between H' and H above, this means that $\mathcal{H}'_\ell(t') = \mathcal{H}_\ell(t'+1)$ for $\ell \in \{0, 1, \ldots, n-1\}$. Therefore, by the induction hypothesis, at the end of any $t \in \{1, \ldots, n\}$, $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| = \left|\bigcup_{\ell=0}^{k} \mathcal{H}'_\ell(t-1)\right| \leq k$, for all $0 \leq k \leq n-1$. For the missing cases, namely $t = 0$ and $0 \leq k \leq n$ or $0 \leq t \leq n$ and $k = n$, we note that they are both trivial: (i) for $t = 0$, any $\mathcal{H}_\ell(0)$ is empty with the possible exception of $\mathcal{H}_n(0)$ that may contain up to n agents, and (ii) for $k = n$, $\left|\bigcup_{\ell=0}^{n} \mathcal{H}_\ell(t)\right| \leq n$ trivially holds for any t. We conclude that at the end of any $t \leq n$, we have $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k$, for all $0 \leq k \leq n$.

Case 2: $v_1(g_1) = \beta_1 \leq \alpha_1$ Since the very first good, g_1, was allocated as a low-valued good despite $\chi_i = 0$, for all $i \in N$, it must be the case that $v_i(g_1) = \beta_i$. That is, $H_1[i] = H_0[i] \geq n$, for all $i \in N$. In this case, the initial vector H'_0 of the sub-instance given as input is defined by $H'_0[i] = H_0[i] - 1 \geq n - 1$ for $i \in \{2, \ldots, n\}$. Like in Case 1, the priority among agents remains exactly the same as in the original instance but here $H'_{t'}[i] = H_{t'+1}[i] - 1$ for all $t' \in \{0, \ldots, n-1\}$ and $i \in \{2, \ldots, n\}$; again, $H'_{t'}[1]$ is not defined.

Unlike Case 1, however, here agent 1 may appear in $\mathcal{H}_\ell(t)$ for several combinations of ℓ and t. Nevertheless, this will not be an issue. First notice that, even if things went really wrong, there are not enough goods for $H'_{t'}[i]$ to become negative for any $i \in \{2, \ldots, n\}$ and any $t' \in \{0, \ldots, n-1\}$, and so $\mathcal{H}'_{-1}(t') = \emptyset$ for all $t' \in \{0, \ldots, n-1\}$. By the correspondence between H' and H discussed above, we have that $\mathcal{H}_\ell(t) \setminus \{1\} = \mathcal{H}'_{\ell-1}(t-1)$ and, thus, $\mathcal{H}_\ell(t) \subseteq \mathcal{H}'_{\ell-1}(t-1) \cup \{1\}$, for $\ell \in \{0, 1, \ldots, n\}$. Therefore, by the induction hypothesis, at the end of any $t \in \{1, \ldots, n\}$,

$$\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq \left|\bigcup_{\ell=0}^{k} \mathcal{H}'_{\ell-1}(t-1) \cup \{1\}\right| \leq \left|\bigcup_{\ell=0}^{k-1} \mathcal{H}'_{\ell}(t-1)\right| + 1 \leq k - 1 + 1 = k\,,$$

for all $0 \leq k \leq n$. The missing cases ($t = 0$ and $0 \leq k \leq n$) are trivial exactly like their counterparts in Case 1. We conclude that at the end of any $t \leq n$, we have $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k$, for all $0 \leq k \leq n$. CL. 4.6 ⊡

It is not hard to see that the proof of Claim 4.6 cannot be extended beyond $t = n$ as it crucially relies on the fact that during Phase 0 (which has fixed length and lasts up to $t = n$) once an agent gets a good, they are inactive for the rest of the phase. Interestingly enough, the proof of Claim 4.7 below (induction with respect to t) could not have been used for $t < n$ as it crucially depends on $H[i]$ not being relevant unless agent i actively competes for a high-valued good is allocated as high-valued (which, in turn, is the result of the "slack" we add to $H[i]$ after an agent i gets a high-valued good).

Claim 4.7. *At the end of any time step $t \geq n$, we have $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k$, for all $0 \leq k \leq n$.*

Proof of Claim 4.7. Given an arbitrary instance, we will prove the statement using strong induction on the time step t. Essentially, Claim 4.6 serves as the base case. So, assume that the statement of the claim is true for all time steps up to a certain $t_0 \geq n$, and consider the next time step $t = t_0 + 1$. Let g be the t-th good and j be the agent who eventually gets it.

Case 1: $v_j(g) = \beta_j \leq \alpha_j$. That is, g is allocated as a low-valued good. We claim that for any agent $i \in N$, such that $H_t[i] \leq n - 1$, we have $H_t[i] = H_{t-1}[i]$, i.e., the entries of H_t may have changed only for agents who are irrelevant with respect to condition (1). Indeed, if $H_t[i]$ has changed, then $v_i(g) = \alpha_i$. Moreover, it must be $\chi_i = 0$, as otherwise $N_h(g, t) \neq \emptyset$ and g would be allocated as a high-valued good instead. But $\chi_i = 0$ means that agent i has received a high-valued good, say at a time step t_h, during the current phase. Recall that each phase has at most $2n - 1$ time steps (see the proof of part 3. of Theorem 4.1) and, thus, $t - t_h \leq 2n - 2$. Also, by the induction hypothesis (for time step t_h and $k = 0$), we have $H_{t_h-1}[i] \geq 1$. Combining these, we get

$$H_t[i] \geq H_{t_h}[i] - (2n - 2) = H_{t_h-1}[i] - 1 + 3n - 2 - 2n + 2 \geq n\,, \tag{3}$$

as claimed. This means that $\mathcal{H}_\ell(t) = \mathcal{H}_\ell(t-1)$ for $\ell \in \{0, 1, \ldots, n-1\}$. Therefore, by invoking the induction hypothesis, at the end of time step t, we have $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| = \left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t-1)\right| \leq k$, for all $0 \leq k \leq n - 1$. For the missing case, namely $k = n$, it is trivial as $\left|\bigcup_{\ell=0}^{n} \mathcal{H}_\ell(t)\right| \leq n$ always holds for any t. We conclude that at the end of time step t, we have $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq k$, for all $0 \leq k \leq n$.

Case 2: $v_j(g) = \alpha_j$. That is, we assume next that g is allocated as a high-valued good. Let $N_h^+(g,t) = \{i \in N \mid v_i(g) = \alpha_i\}$, i.e., the set of all agents who see the t-th good as high-valued. The agents of $N_h^+(g,t)$ are exactly those whose entries in H_t are updated during the current time step and how relevant it is for our analysis. We carefully categorize agents according to what happens to their entry in H_t during time step t:

- For any agent $i \in N \setminus N_h^+(g,t)$, $H_t[i] = H_{t-1}[i]$, as they see g as a low-valued good.
- For any agent $i \in N_h(g,t) \setminus \{j\}$, $H_t[i] = H_{t-1}[i] - 1$, as they see g as a high-valued good and they miss it.
- For any agent $i \in N_h^+(g,t) \setminus N_h(g,t)$, again $H_t[i] = H_{t-1}[i] - 1$, as they see g as a high-valued good and they miss it, but they are essentially irrelevant because $H_t[i] \geq n$. This follows by the argument in Case 1 above, as $\chi_i = 0$ and the chain of (in)equalities of (3) applies exactly as is.
- For agent j itself, we have $H_t[j] = H_{t-1}[j] - 1 + 3n - 2 \geq 3n - 2 \geq n$, where $H_{t-1}[j] \geq 1$ follows from the induction hypothesis for time step $t-1$ and $k = 0$. We conclude that agent j is also essentially irrelevant.

From the above, it becomes clear that agents in $N_h(g,t) \cap \bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)$ should be treated carefully when we try to bound $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right|$, for some $k \in \{0, 1, \ldots, n\}$. The easiest case, of course, is when $N_h(g,t) \cap \bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t) = \emptyset$. Then, $\mathcal{H}_\ell(t) = \mathcal{H}_\ell(t-1)$ for $\ell \leq k$ and, thus, by the induction hypothesis, we have that at the end of time step t, $\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| = \left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t-1)\right| \leq k$. Also, when $k = n$, it trivially holds $\left|\bigcup_{\ell=0}^{n} \mathcal{H}_\ell(t)\right| \leq n$.

Next, assume that $k \in \{0, 1, \ldots, n-1\}$ is such that $N_h(g,t) \cap \bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t) = S \neq \emptyset$. At this point, we need three simple observations. The first one (following from the last bullet above) is that $j \notin \bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)$ and, thus, $j \notin S$. The second one is that $\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t) \subseteq \bigcup_{\ell=0}^{k+1} \mathcal{H}_\ell(t-1)$, as no entry of H reduces by more than 1 in a single time step. The last one is that $j \in \bigcup_{\ell=0}^{k+1} \mathcal{H}_\ell(t-1)$; if not, we would have $H_{t-1}[j] \geq k+2$ and the j-th entry of H right before g was allocated to j would be $H_{t-1}[j] - 1 \geq k+1 > k \geq \min_{i \in S} H[i] \geq \min_{i \in N_h(g,t)} H[i]$, contradicting the choice of j. Using these observations, as well as the induction hypothesis, we have that at the end of time step t,

$$\left|\bigcup_{\ell=0}^{k} \mathcal{H}_\ell(t)\right| \leq \left|\bigcup_{\ell=0}^{k+1} \mathcal{H}_\ell(t-1) \setminus \{j\}\right| = \left|\bigcup_{\ell=0}^{k+1} \mathcal{H}_\ell(t-1)\right| - 1 \leq (k+1) - 1 = k.$$

This exhausts all possible cases for $k \in \{0, 1, \ldots, n\}$ and concludes Case 2. Cl. 4.7 ⊡

Clearly, combining the two claims completes the proof of the lemma. □

5 The Power of Limited Foresight

A natural question at this point is whether one can avoid the linear term in the MMS approximation guarantee by allowing some additional information about the future. Unfortunately, there is a very simple instance that illustrates that this is not possible.

Proposition 5.1. *Let $\varepsilon > 0$ be a constant. Even if the whole instance is known in advance, as long as it is required to irrevocably allocate each good right after it arrives, no algorithm can always compute a $(1/n + \varepsilon)$-temporal-MMS allocation, even on 2-value instances.*

Proof. We consider a 2-value instance with n agents, where $\beta_i = 1$ and $\alpha_i = \alpha \geq n$, for all $i \in N$. There are n universally low-valued goods $g_1, \ldots, g_n$ and $n-1$ universally high-valued goods $g_{n+1}, \ldots, g_{2n-1}$, as shown below, that arrive according to their indices.

	g_1	g_2	$\cdots$	g_{n-1}	g_n	g_{n+1}	g_{n+2}	$\cdots$	g_{2n-1}
ag. 1:	(1)	1	…	1	1	α	α	…	α
ag. 2:	1	(1)	…	1	1	α	α	…	α
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋱	⋮
ag. $n-1$:	1	1	…	(1)	1	α	α	…	α
ag. n:	1	1	…	1	(1)	α	α	…	α

We may assume that this information is available even before the first good arrives. Like in the proof of Theorem 3.2, we note that if at any point during the first n time steps an agent receives a second good, then at the end of time step $t = n$ there would be at least one agent that has received value 0 despite having a positive maximin share value. So, we may assume that any algorithm with non-trivial guarantees assigns to each agent exactly one of the first n goods. Without loss of generality, agent i gets good g_i as shown. Given that, no matter how goods $g_{n+1}, \ldots, g_{2n-1}$ are allocated, at least one agent, say agent 1, will not receive any of these. So, at the end of time step $t = 2n-1$ we have $v_i(A_1) = v_1(g_1) = 1$, while $\boldsymbol{\mu}_1(2n-1) = n$. □

Despite Proposition 5.1, in this section we show that being able to see only a linear number of steps in the future leads to significantly simpler algorithms with EF1 and EF2 guarantees.

5.1 The Illustrative Case of Two Agents

There is an easy algorithm augmented with foresight of length 1 that achieves EF1 in every *even* time step. Although Naive-Matching (Algorithm 2) is essentially subsumed by the main result of the next section, it is simpler to state, much simpler to analyze and still illustrates the power of—even very limited—foresight.

Theorem 5.2. *For any two agent 2-value instance augmented with foresight of length 1, Algorithm 2 builds an allocation that is temporal-EF2, while it is EF1 for every* even *time step $t \geq 0$.*

Proof. We first observe that if an allocation is EF1 at the end of some time step $t \geq 0$, then it will trivially be EF2 at the end of time step $t+1$, no matter how the corresponding good is allocated. That is, it suffices to show that Algorithm 2 builds an allocation that is EF1 for every *even* time step t, and the fact that it is also temporal-EF2 immediately follows. We will show a somewhat stronger statement for even time steps: *for any $k \in \mathbb{Z}_{\geq 0}$, at the end of time step $t = 2k \leq m$,* (i) *both bundles contain k goods each, and* (ii) *either* `ctr` $= 0$ *and only agent 1 may envy agent 2 by at most $\alpha_1 - \beta_1$, or* `ctr` $= 1$ *and only agent 2 may envy agent 1 by at most $\alpha_2 - \beta_2$.* Of course, (i) is straightforward because, for any $k \geq 0$, we always create a matching between the two agents and the goods g_{2k+1}, g_{2k+2}. We are going to show (ii) using induction on k.

At time $t = 0$ (i.e., for $k = 0$) the statement of (ii) trivially holds: `ctr` $= 0$ and no agent envies the other. Now assume that (ii) holds for some $t = 2k$, such that $k \geq 0$ and $t = 2k+2 \leq m$. At time $t = 2k+1$ the algorithm enters the 'else' in line 8. Note that whenever the condition of line 10 is true, i.e., goods g_{2k+1}, g_{2k+2} induce any pattern of Table 1 except of those of blocks I or II, each agent (weakly) prefers the good they receive according to the corresponding allocation shown in Table 1. That is, for agent 1 we have $v_1(A_1^{2k+2}) - v_1(A_1^{2k}) \geq v_1(A_2^{2k+2}) - v_1(A_2^{2k})$, where the added superscript indicates the time step at the end of which we consider the bundle, and similarly for agent 2. This observation that envy can only be

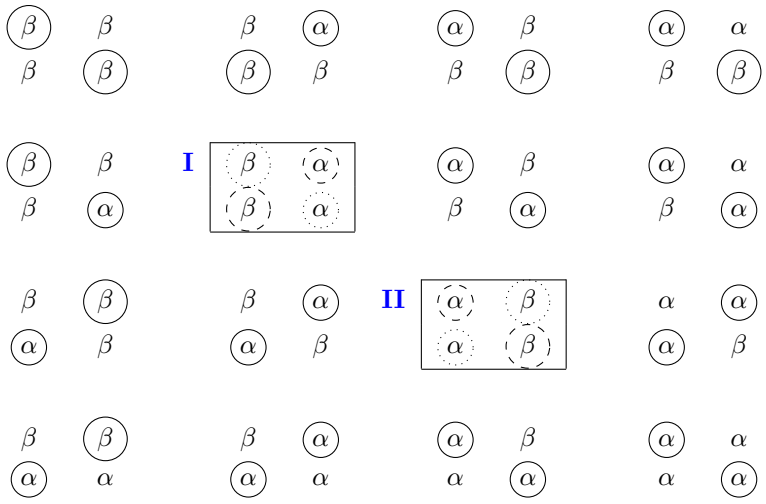

Table 1: When we are given foresight of length 1, achieving EF1 on every other time step for two agents is very simple. We only need to keep track of who "wins" the next block of type I or II, i.e., who gets the contested high-valued good (*dashed:* agent 1 wins, *dotted:* agent 2 wins). In every other case, we may allocate the goods in a predetermined way as shown for any other block here. Since we only care about the general value patterns, for the sake of readability, we omit the indices; e.g., we write α rather than α_1 in the first row and α_2 in the second row of each block.

reduced in this case, combined with the fact that `ctr` does not change here, implies that (ii) still holds for $t = 2k + 2$.

So, we may assume that the condition of line 10 is false, i.e., goods g_{2k+1}, g_{2k+2} induce the pattern of block I or block II of Table 1 with one universally high-valued and one universally low-valued good. Here we consider the two cases of the induction hypothesis.

Case 1: at $t = 2k$, `ctr` $= 0$, $v_1(A_1^{2k}) \geq v_1(A_2^{2k}) - (\alpha_1 - \beta_1)$ and $v_2(A_2^{2k}) \geq v_2(A_1^{2k})$. According to lines 14-17, in this case, `ctr` becomes 1 and the algorithm commits to giving the high-valued good to agent 1. Thus,

$$v_1(A_1^{2k+2}) = v_1(A_1^{2k}) + \alpha_1 \geq v_1(A_2^{2k}) - (\alpha_1 - \beta_1) + \alpha_1 = v_1(A_2^{2k+2}) \,,$$

as well as

$$v_2(A_2^{2k+2}) = v_2(A_2^{2k}) + \beta_2 \geq v_2(A_1^{2k}) + \beta_2 = v_2(A_1^{2k+2}) - \alpha_2 + \beta_2 \,,$$

i.e., (ii) still holds for $t = 2k + 2$.

Case 2: at $t = 2k$, `ctr` $= 1$, $v_1(A_1^{2k}) \geq v_1(A_2^{2k})$ and $v_2(A_2^{2k}) \geq v_2(A_1^{2k}) - (\alpha_2 - \beta_2)$. This is completely symmetric to Case 1 above. According to lines 14-17, `ctr` becomes 0 and the algorithm commits to giving the high-valued good to agent 2. Thus,

$$v_1(A_1^{2k+2}) = v_1(A_1^{2k}) + \beta_1 \geq v_1(A_2^{2k}) + \beta_1 = v_1(A_2^{2k+2}) - \alpha_1 + \beta_1 \,,$$

and

$$v_2(A_2^{2k+2}) = v_2(A_2^{2k}) + \alpha_2 \geq v_2(A_1^{2k}) - (\alpha_2 - \beta_2) + \alpha_2 = v_2(A_1^{2k+2}) \,,$$

i.e., (ii) still holds for $t = 2k + 2$.

This concludes the induction and shows that Algorithm 2 builds an allocation that is EF1 for every even time step t and, thus, temporal-EF2. □

Algorithm 2 Naive-Matching$(v_1, v_2; M)$

(The valuation functions, v_1, v_2, are given via oracles; M is given in an online fashion, one good at a time, along with a preview of the next good after that.)

$t \leftarrow 0$; `ctr` $\leftarrow 0$ // We initialize our counters and the allocation.
for $i \in N$ **do**
    $A_i \leftarrow \emptyset$
whenever a new good g **arrives along with a preview of the next good,** g'**:**
$t \leftarrow t + 1$
if $t = 0 \bmod 2$ **then**
    Allocate g according to the commitment of time step $t - 1$.
else
    $B \leftarrow \begin{bmatrix} v_1(g) & v_1(g') \\ v_2(g) & v_2(g') \end{bmatrix}$
    if B follows any pattern of Table 1 except of those of blocks I or II **then**
        Add g to A_1 or A_2 according to the allocation of the corresponding pattern in Table 1.
        Commit to add g' to A_1 or A_2 at time step $t + 1$ according to the aforementioned allocation.
    else
        `ctr` $\leftarrow$ (`ctr` $+ 1) \bmod 2$
        $j = 2 -$ `ctr`
        Add g to A_1 or A_2 according to the allocation of the corresponding pattern in Table 1 (i.e., I or II) which gives the contested high-valued good to agent j.
        Commit to add g' to A_1 or A_2 at time step $t + 1$ according to the aforementioned allocation.

Corollary 5.3. *Assuming m is large enough, for any $\lambda \in \mathbb{Z}_{>0}$, after a sufficient number of steps, the allocation built by Algorithm 2 becomes and remains $\lambda/(\lambda+2)$-EF, $\lambda/(\lambda+1)$-EF1, and $\lambda/(\lambda+1)$-PROP.*

Proof. Clearly, if m is large enough (e.g., $m \geq 2\lambda \max_{i\in[2]} \lceil \alpha_i \rceil$) there is some t_* by which each agent i has received value equal to at least $\lambda\alpha_i$. At the end of any time step $t \geq t_*$, we have for agent 1

$$v_1(A_2^t) \leq \min_{g,g' \in A_2^t} v_1(A_2^t \setminus \{g, g'\}) + 2\alpha_1 \leq v_1(A_1^t) + 2\alpha_1 \leq (1 + 2/\lambda) v_1(A_1^t)\,,$$

$$\min_{g \in A_2^t} v_1(A_2^t \setminus \{g\}) \leq \min_{g,g' \in A_2^t} v_1(A_2^t \setminus \{g, g'\}) + \alpha_1 \leq v_1(A_1^t) + \alpha_1 \leq (1 + 1/\lambda) v_1(A_1^t)\,, \quad \text{and}$$

$$0.5(v_1(A_1^t) + v_1(A_2^t)) \leq 0.5\big[v_1(A_1^t) + (1 + 2/\lambda) v_1(A_1^t)\big] = (1 + 1/\lambda) v_1(A_1^t)\,,$$

and similarly for agent 2. □

5.2 Foresight of Length $n - 1$ Suffices

As we mentioned in the beginning of the previous section, we can generalize Theorem 5.2 to any number of agents, albeit with a more complicated algorithm.

Similarly to Algorithm 2, what we would like to achieve between time steps kn and $(k + 1)n$ (i.e., with goods $g_{kn+1}, \ldots, g_{(k+1)n}$) is to obtain an EF1 allocation by taking the union of the EF1 allocation of time step kn and an appropriately chosen matching. However, it is easy to see that this fails if not done carefully, even for two agents, which is the reason why Algorithm 2 treats the patterns of blocks I and II with some care. In order to achieve a similar thing for general n, at time step $kn + 1$ we construct a matching $\mathcal{M}$ involving the current good and the $n - 1$ predicted goods, so that the matching's envy graph "cancels

out" any problematic edges of the envy graph of the EF1 allocation of time step kn. Then, for what we call the $(k+1)$-th round, we allocate these n goods according to $\mathcal{M}$. For $\mathcal{M}$ to have the nice aforementioned behavior, however, we take it to be a maximum weight matching with respect to carefully defined auxiliary valuation functions. These auxiliary functions encode all the information about which goods are high or low for which agents, while giving additional weight to goods for agents who come earlier in the topological sorting. The latter captures the intuition that such agents should have a higher priority during this round.

Algorithm 3 Priority-Matching$(v_1, \ldots, v_n; M)$

(The valuation functions, $v_i, i \in [n]$, are given via oracles; M is given in an online fashion, one good at a time, along with a preview of the next $n-1$ goods after that.)

$t \leftarrow 0$; `ctr` $\leftarrow 0$ // We initialize our counters and the allocation.
for $i \in N$ **do**
    $A_i \leftarrow \emptyset$
whenever a new good g **arrives along with a preview of the next** $n-1$ **goods**
$t \leftarrow t+1$
if $t = 1 \bmod n$ **then** // A new *round* begins, namely round number $\lceil t/n \rceil$.
    Construct the current envy-graph $G_t = (V_t, E_t)$. // $V_t = N$ and $(i,j) \in E_t$ if $v_i(A_j) > v_i(A_i)$.
    Find a permutation π that induces a topological sorting of G_t. // If $(i,j) \in E_t$, then $\pi(i) < \pi(j)$.
    for $i \in [n]$ and $h \in \{g_t, g_{t+1}, \ldots, g_{t+n-1}\}$ **do** // Define auxiliary valuation functions consistent with π.
        $\tilde{v}_{\pi^{-1}(i)}(h) \leftarrow \begin{cases} 2(1+1/2n)^{n-i} & \text{if } v_{\pi^{-1}(i)}(h) = \alpha_{\pi^{-1}(i)} \\ (1+1/2n)^{n-i} & \text{otherwise} \end{cases}$ // $\pi^{-1}(i)$ is the i-th agent in the ordering.
    Find a maximum weight matching $\mathcal{M}$ between the agents and goods in $\{g_t, \ldots, g_{t+n-1}\}$ with respect to these auxiliary functions. // The weight of a pair (j,h) is $\tilde{v}_j(h)$. The matching is perfect unless this is the last round; in this case, the matching is between N and $\{g_t, \ldots, g_m\}$.
    Add $g = g_t$ to a bundle according to $\mathcal{M}$, i.e., add g_t to A_j if and only if $(j, g_t) \in \mathcal{M}$.
    Commit to also allocate $\{g_{t+1}, \ldots, g_{t+n-1}\}$ according to $\mathcal{M}$ at time steps $t+1$ through $t+n-1$.
else // The allocation of g was decided at the beginning of this round.
    Allocate g according to the commitment of time step $(\lceil t/n \rceil - 1)n + 1$ (when this round begun).

Theorem 5.4. *For any personalized 2-value instance augmented with foresight of length $n-1$, Algorithm 3 builds an allocation that is temporal-EF2, while it is EF1 (and, thus, $1/n$-MMS) for every time step $t = kn, k \in \mathbb{Z}_{\geq 0}$. Moreover, if at any step t_0 the allocation fails to be $1/2$-EF1, then it remains $1/2$-EF1 at the end of every time step $t \geq \lceil t_0/n \rceil n$.*

Proof. The proof has similar structure to the proof of Theorem 5.2, but the induction is fairly more complicated due to the fact that the matching process is much less trivial here. In what follows, we refer to the execution of Algorithm 3 for time steps $kn+1, kn+1, \ldots, (k+1)n$ as the $(k+1)$-th *round*, for any $k \in \mathbb{Z}_{\geq 0}$.

We begin with a generalization of the first observation made in the proof of Theorem 5.2: if an allocation is EF1 at the end of some time step $t \geq 0$, then it will trivially be EF2 at the end of time step $t+\ell$, as long as no bundle receives more than one out of the last ℓ goods. For Algorithm 3, this means that if the allocation is EF1 at the end of time step $t = kn$ (i.e., right before round $k+1$ begins), for every $k \in \mathbb{Z}_{\geq 0}$, then it remains EF2 for every time step in between. This holds just because in every round each agent receives (at most) one good via the corresponding matching $\mathcal{M}$. So, it suffices to show that Algorithm 3 builds an allocation that is EF1 for every time step t which is a multiple of n, and the fact that it is also temporal-EF2 immediately follows. Again we show a somewhat stronger statement: *for any $k \in \mathbb{Z}_{\geq 0}$, at the end of time step $t = kn \leq m$,*

(i) all bundles contain k goods each;

(ii) the corresponding envy graph (i.e., G_{kn+1} constructed in the beginning of the next time step in line 7) is acyclic;

(iii) any edge (i, j) in the above graph indicates envy which is at most $\alpha_i - \beta_i$.

Like before, (i) is straightforward, since the algorithm always creates a matching between the agents and (at most) n goods in the beginning of a round and, as these goods arrive, it allocates them according to the matching. We are going to show (ii) and (iii) using induction on k. At time $t = 0$ (i.e., for $k = 0$) the statements of (ii) and (iii) trivially hold, as no agent envies another agent. Now assume that (ii) and (iii) hold at the end of some time step $t = kn$, such that $k \geq 0$ and $t = (k + 1)n \leq m$. At the beginning of time step $t = kn + 1$ the algorithm enters the 'if' in line 6 and constructs the envy graph $G := G_{kn+1}$ in line 7. This G is exactly the envy graph for which the two parts of the induction hypothesis hold. The fact that G is acyclic is what allows it to be topologically sorted (see, e.g., Cormen et al. [2022]) and, hence, makes line 8 well-defined. We use G' to denote the envy graph at the end of the current (i.e., the $(k + 1)$-th) round; note that G' is constructed as $G_{(k+1)n+1}$ at the beginning of the next round.

Claim 5.5. *Any edge (i, j) of G' indicates envy which is at most $\alpha_i - \beta_i$.*

Proof of Claim 5.5. Consider any edge (i, j) of G' and let h_i, h_j be the goods agents i and j received, respectively, in round $k + 1$. This means that h_i, h_j were matched with i and j, respectively, in $\mathcal{M}$. By the induction hypothesis, we know that the envy from agent i towards agent j at the end of time step $t = kn$—if it existed at all—was upper bounded by $\alpha_i - \beta_i$, i.e., $v_i(A_j^{kn}) - v_i(A_i^{kn}) \leq \alpha_i - \beta_i$. If $v_i(h_i) \geq v_i(h_j)$, then

$$v_i(A_j^{(k+1)n}) - v_i(A_i^{(k+1)n}) = v_i(A_j^{kn}) + v_i(h_j) - v_i(A_i^{kn}) - v_i(h_i) \leq \alpha_i - \beta_i \,.$$

We need to consider the case where $\beta_i = v_i(h_i) < v_i(h_j) = \alpha_i$. If there was no edge (i, j) in G, i.e., if $v_i(A_j^{kn}) - v_i(A_i^{kn}) \leq 0$, then clearly

$$v_i(A_j^{(k+1)n}) - v_i(A_i^{(k+1)n}) = v_i(A_j^{kn}) + \alpha_i - v_i(A_i^{kn}) - \beta_i \leq \alpha_i - \beta_i \,.$$

So we may assume that (i, j) was an edge in G. Dy definition, this means that in any topological sorting agent i must come before agent j. In particular, $\hat{\imath} = \pi(i) < \pi(j) = \hat{\jmath}$, i.e., in the sorting induced by the permutation π, agent i is the $\hat{\imath}$-th agent and agent j is the $\hat{\jmath}$-th. By the definition of the auxiliary functions in line 10, we have

$$\tilde{v}_i(h_j) = 2\Big(1 + \frac{1}{2n}\Big)^{n-\hat{\imath}} \quad \text{and} \quad \tilde{v}_i(h_i) = \Big(1 + \frac{1}{2n}\Big)^{n-\hat{\imath}} .$$

Let $\mathcal{M}'$ be the matching that one gets from $\mathcal{M}$ by switching h_i and h_j. That is, in $\mathcal{M}'$ agent i is matched with h_j, agent j is matched with h_i, and every other agent ℓ is matched with h_ℓ as in $\mathcal{M}$. Now, if we use $w(\mathcal{M})$ to denote the sum of weights of the pairs in $\mathcal{M}$, we have

$$\begin{aligned}
w(\mathcal{M}') - w(\mathcal{M}) &= \Big(\tilde{v}_i(h_j) + \tilde{v}_j(h_i) + \sum_{\ell \in N \setminus \{i,j\}} \tilde{v}_\ell(h_\ell)\Big) - \sum_{\ell \in N} \tilde{v}_\ell(h_\ell) \\
&= \tilde{v}_i(h_j) - \tilde{v}_i(h_i) + \tilde{v}_j(h_i) - \tilde{v}_j(h_j) \\
&\geq 2\Big(1 + \frac{1}{2n}\Big)^{n-\hat{\imath}} - \Big(1 + \frac{1}{2n}\Big)^{n-\hat{\imath}} + \Big(1 + \frac{1}{2n}\Big)^{n-\hat{\jmath}} - 2\Big(1 + \frac{1}{2n}\Big)^{n-\hat{\jmath}} \\
&= \Big(1 + \frac{1}{2n}\Big)^{n-\hat{\imath}} - \Big(1 + \frac{1}{2n}\Big)^{n-\hat{\jmath}} = \Big(1 + \frac{1}{2n}\Big)^{n-\hat{\jmath}} \Big[\Big(1 + \frac{1}{2n}\Big)^{\hat{\jmath}-\hat{\imath}} - 1\Big] \\
&\geq \Big(1 + \frac{1}{2n}\Big)^{0} \Big[\Big(1 + \frac{1}{2n}\Big)^{1} - 1\Big] = \frac{1}{2n} > 0 \,,
\end{aligned}$$

where for the first inequality we used the exact values of $\tilde{v}_i(h_j), \tilde{v}_i(h_i)$ and lower and upper bounds for $\tilde{v}_j(h_i), \tilde{v}_j(h_j)$. This, however, contradicts the choice of $\mathcal{M}$ as a maximum weight matching. Thus, under the assumption that $\beta_i = v_i(h_i) < v_i(h_j) = \alpha_i$, (i,j) cannot be an edge in G. We conclude that, in any case, an edge (i,j) of G' indicates envy which is no more than $\alpha_i - \beta_i$. Cl. 5.5 ⊡

Claim 5.6. *The graph G' is acyclic.*

Proof of Claim 5.6. Suppose, towards a contradiction, that G' contains a simple directed cycle $C = (i_1, i_2, \ldots, i_s, i_1)$. Also, let $h_{i_1}, \ldots, h_{i_s}$ denote the goods these agents received, respectively, in round $k+1$. Since G was acyclic, not all edges of C existed in G. Without loss of generality (as it is a matter of renaming the agents / vertices), we may assume that the (i_1, i_2) was not an edge in G. Our first observation is that the only way this could happened, is that $v_{i_1}(h_{i_1}) = \beta_{i_1}$ but $v_{i_1}(h_{i_2}) = \alpha_{i_1}$.

We next note that it must be the case that $v_{i_2}(h_{i_2}) = \alpha_{i_2}$, otherwise we could define a new matching $\mathcal{M}'$ by only switching h_{i_1} and h_{i_2} in $\mathcal{M}$ and improve the maximum weight, similarly to what we did in the proof of Claim 5.5:

$$\begin{aligned} w(\mathcal{M}') - w(\mathcal{M}) &= \tilde{v}_{i_1}(h_{i_2}) - \tilde{v}_{i_1}(h_{i_1}) + \tilde{v}_{i_2}(h_{i_1}) - \tilde{v}_{i_2}(h_{i_2}) \\ &\geq 2\Big(1+\frac{1}{2n}\Big)^{n-\pi(i)} - \Big(1+\frac{1}{2n}\Big)^{n-\pi(i)} + \Big(1+\frac{1}{2n}\Big)^{n-\pi(j)} - \Big(1+\frac{1}{2n}\Big)^{n-\pi(j)} \\ &\geq \Big(1+\frac{1}{2n}\Big)^{0} > 0\,, \end{aligned}$$

where we used $v_{i_2}(h_{i_2}) = \beta_{i_2}$ for the first inequality, contradicting the choice of $\mathcal{M}$. So, it must be $v_{i_2}(h_{i_2}) = \alpha_{i_2}$.

The third observation we need for the proof is that for any edge (i_r, i_{r+1}) in C, such that $v_{i_r}(h_{i_r}) = \alpha_{i_r}$, it must also be the case that $v_{i_r}(h_{i_{r+1}}) = \alpha_{i_r}$.[1] To see this, notice that

$$\begin{aligned} v_{i_r}(A_{i_{r+1}}^{(k+1)n}) - v_{i_r}(A_{i_r}^{(k+1)n}) &= v_{i_r}(A_{i_{r+1}}^{kn}) - v_{i_r}(A_{i_r}^{kn}) + v_{i_r}(h_{i_{r+1}}) - v_{i_r}(h_{i_r}) \\ &\leq \alpha_{i_r} - \beta_{i_r} + v_{i_r}(h_{i_{r+1}}) - \alpha_{i_r} \\ &= v_{i_r}(h_{i_{r+1}}) - \beta_{i_r}\,, \end{aligned}$$

where we used the induction hypothesis for the first inequality. Since this difference must be positive for (i_r, i_{r+1}) to be in C, we get that $v_{i_r}(h_{i_{r+1}}) = \alpha_{i_r}$.

Finally, we distinguish two cases, depending on whether there is another agent, besides i_1, who sees the good it received in this round as low-valued.

Case 1: $v_{i_r}(h_{i_r}) = \alpha_{i_r}$, for all $r \in \{2, \ldots, s\}$. In this case, we can get a new matching $\mathcal{M}'$ by assigning each good among $h_{i_1}, \ldots, h_{i_s}$ to the 'previous' agent with respect to the cycle, i.e., match h_{i_1} to i_s, h_{i_2} to i_1, and so on, and keeping the rest of the matching the same as $\mathcal{M}$. By the third observation above, we have that $v_{i_r}(h_{i_{r+1}}) = \alpha_{i_r}$, for all $r \in \{2, \ldots, s\}$, whereas by our first observation about agent i_1's envy, we also have $v_{i_1}(h_{i_2}) = \alpha_{i_1}$. Note that no weight is decreased going from $\mathcal{M}$ to $\mathcal{M}'$ and $\tilde{v}_{i_1}(h_{i_2})$ is now increased to $2\big(1+\frac{1}{2n}\big)^{n-\pi(i_1)}$ (from $\big(1+\frac{1}{2n}\big)^{n-\pi(i_1)}$ that was in $\mathcal{M}$). This contradicts the choice of $\mathcal{M}$ as a maximum weight matching.

Case 2: $v_{i_r}(h_{i_r}) = \beta_{i_r}$, for some $r \in \{2, \ldots, s\}$. Let ℓ the smallest such r. That is, $v_{i_x}(h_{i_x}) = \alpha_{i_x}$, for all $x \in \{2, \ldots, \ell-1\}$, but $v_{i_\ell}(h_{i_\ell}) = \beta_{i_\ell}$. In this case, we can get a new matching $\mathcal{M}'$ by assigning each good among $h_{i_1}, \ldots, h_{i_\ell}$ to the 'previous' agent with respect to the cycle and , i.e., match h_{i_2} to i_1, h_{i_3} to i_2,

[1] We use the standard convention that $i_{s+1} := i_1$.

and so on, as well as h_{i_1} to i_ℓ, while keeping the rest of the matching the same as $\mathcal{M}$. Now we can argue like in Case 1. We have $v_{i_r}(h_{i_{r+1}}) = \alpha_{i_r}$, for all $r \in \{2, \dots, \ell - 1\}$, by our third observation, whereas $v_{i_1}(h_{i_2}) = \alpha_{i_1}$ like before and, of course, $v_{i_\ell}(h_{i_1}) \geq \beta_{i_\ell}$. Again, no weight is decreased going from $\mathcal{M}$ to $\mathcal{M}'$ and $\tilde{v}_{i_1}(h_{i_2})$ is increased from $\left(1 + \frac{1}{2n}\right)^{n-\pi(i_1)}$ to $2\left(1 + \frac{1}{2n}\right)^{n-\pi(i_1)}$ (possibly ℓ's weight increased as well). Like in Case 1, this contradicts the choice of $\mathcal{M}$ as a maximum weight matching.

We conclude that G' cannot contain any cycles. Cl. 5.6 ⊡

Claims 5.5 and 5.6 complete the induction. Thus, Algorithm 3 builds an allocation that is EF1 for every time step which is a multiple of n and, because the allocation is balanced, it is also temporal-EF2.

What remains to be shown is the last part of the theorem. Suppose that at the end of some step t_0 the allocation fails to be $1/2$-EF1. Note that this happens during round $k = \lceil t_0/n \rceil$ and let i, j be two agents, such that $v_i(A_i^{t_0}) < 0.5 v_i(A_j^{t_0} \setminus S)$ for any $S \subseteq A_j^{t_0}$ with $|S| \leq 1$. It is easy to see that this can only happen if i was envious of j already at the end of round $k-1$; otherwise no single good added in j's bundle can violate (even exact) EF1 from i's perspective. Besides this, we also claim that, if h_i, h_j are the goods agents i and j receive, respectively, in round k, then $v_i(h_j) = \alpha_i$. Indeed, using property (iii) we showed above, even if agent j receives its good first in round k, we have

$$v_i(A_j^{t_0}) - v_i(A_i^{t_0}) \leq v_i(A_j^{(k-1)n}) + v_i(h_j) - v_i(A_i^{(k-1)n}) \leq \alpha_i - \beta_i + v_i(h_j)\,.$$

So, if h_j was low-valued for agent i, we would be able to eliminate i's envy towards j by just removing a high-valued good from $A_j^{t_0}$ (which must exist, otherwise $v_i(A_j^{t_0})$ and $v_i(A_i^{t_0})$ would only differ by a single low-valued good at most). Now, given that $v_i(h_j) = \alpha_i$ and that i was already envious of j, it must also be the case that $v_i(h_i) = \alpha_i$, or we could repeat the exact same argument as in the proof of Claim 5.5 (i.e., construct the matching $\mathcal{M}'$ by switching h_i and h_j and get a contradiction by showing it has larger weight than $\mathcal{M}$). Therefore, by the end of the round (at time step kn), we have $v_i(A_i^{kn}) \geq \alpha_i$. Since the allocation is temporal-EF2, at the end of any time step $t \geq kn$, we have

$$v_i(A_i^t) \geq \min_{S:|S|\leq 2} v_i(A_j^t \setminus S) \geq \min_{S:|S|\leq 1} v_i(A_j^t \setminus S) - \alpha_i \geq \min_{S:|S|\leq 1} v_i(A_j^t \setminus S) - v_i(A_i^t)\,,$$

and, thus, $v_i(A_i^t) \geq 0.5 \min_{S:|S|\leq 1} v_i(A_j^t \setminus S)$, i.e., the allocation at the end of time step t is $1/2$-EF1 □

Like in Section 5.1, we can get the analog of Corollary 5.3 but for any number of agents.

Corollary 5.7. *For any $\lambda \in \mathbb{Z}_{>0}$, if m is large enough, after a sufficient number of steps the allocation built by Algorithm 3 becomes and remains $\lambda/(\lambda+2)$-EF, $\lambda/(\lambda+1)$-EF1, and $\lambda/(\lambda+2)$-PROP.*

Proof. The proof differs from that of Corollary 5.3 only in getting the guarantee for proportionality. Assuming that by time step t_* by each agent i has received value equal to at least $\lambda\alpha_i$, we have for agent 1

$$\frac{1}{n}\sum_{i=1}^{n} v_1(A_i^t) \leq \frac{1}{n}\left[v_1(A_1^t) + (n-1)\left(1 + \frac{2}{\lambda}\right)v_1(A_1^t)\right] = \left(1 + \frac{2(n-1)}{\lambda n}\right)v_1(A_1^t) \leq \left(1 + \frac{2}{\lambda}\right)v_1(A_1^t)\,,$$

and similarly for agents 2 through n. □

6 Going Beyond 2-Value Instances

As we mentioned in the Introduction, there is a simple way of approximating any additive instance via a 2-value instance: we set a threshold for each agent and we round everything up to the maximum value this

agent has for any good or down to the minimum value this agent has for any good. Of course, this naive idea does not give any guarantees, in general, but it seems like it is a very natural approach for personalized interval-restricted instances. Indeed, we can transfer all of our positive results to personalized interval-restricted instances at the expense of an additional multiplicative factor that is equal to the harmonic mean of the upper and lower bounds of the values an agent may have for the goods.

Theorem 6.1. *For any personalized interval-restricted instance (augmented with foresight or not), there is a simple reduction to a personalized 2-value instance (which is equally augmented), so that any guarantee with respect to EF, EF1, EF2, PROP, or MMS we may obtain for the latter (e.g., via Algorithms 1, 2, or 3) can be translated to the same guarantee for the original instance at the expense of an additional multiplicative factor of $\sqrt{\alpha_i}$ for each agent i.*

Proof. The main idea is to use auxiliary valuation functions that aim to approximate the original valuation functions while taking only two values. Given the personalized interval-restricted valuation function v_i of an agent i, such that $v_i(g) \in [1, \alpha_i]$, for all $g \in M$, we define the personalized 2-value *threshold* function $\hat{v}_i$ as follows:

$$\hat{v}_i(g) = \begin{cases} \alpha_i\,, & \text{if } v_i(g) > \sqrt{\alpha_i} \\ \sqrt{\alpha_i}\,, & \text{otherwise} \end{cases}$$

for any $g \in M$. It is not hard to see how v_i and $\hat{v}_i$ are related.

Claim 6.2. *For any set of goods $S \subseteq M$ and any agent $i \in N$, it holds that $\hat{v}_i(S)/\sqrt{\alpha_i} \leq v_i(S) \leq \hat{v}_i(S)$.*

Proof of Claim 6.2. For any $g \in M$, $v_i(g)$ is rounded up in order to obtain $\hat{v}_i(g)$, so it is straightforward that $v_i(g) \leq \hat{v}_i(g)$. On the other hand, $v_i(g)$ is always rounded up by a factor that is at most $\sqrt{\alpha_i}$, so $\sqrt{\alpha_i}\, v_i(g) \geq \hat{v}_i(g)$. Since both functions are additive, these inequalities extend to any set of goods. Cl. 6.2 ⊡

We first argue about EF, EF1, and EF2. Suppose that at the end of some time step t the allocation $(A_1^t, \ldots, A_n^t)$ is ρ-EFk (where EF0 is just EF) with respect to the threshold functions $\hat{v}_1, \ldots, \hat{v}_n$. Then, for any $i, j \in N$, we have

$$v_i(A_i^t) \geq \frac{1}{\sqrt{\alpha_i}}\, \hat{v}_i(A_i^t) \geq \frac{1}{\sqrt{\alpha_i}}\, \rho \min_{S:|S|\leq k} \hat{v}_i(A_j^t \setminus S) \geq \frac{\rho}{\sqrt{\alpha_i}} \min_{S:|S|\leq k} v_i(A_j^t \setminus S)\,,$$

where the first and the third inequalities follow from Claim 6.2. Thus, $(A_1^t, \ldots, A_n^t)$ is $\rho/\sqrt{\alpha_i}$-EFk with respect to the original functions $v_1, \ldots, v_n$.

Next, suppose that at the end of time step t the allocation $(A_1^t, \ldots, A_n^t)$ is ρ-PROP with respect to $\hat{v}_1, \ldots, \hat{v}_n$. Then, similarly to the above, for any $i \in N$, we have

$$v_i(A_i^t) \geq \frac{1}{\sqrt{\alpha_i}}\, \hat{v}_i(A_i^t) \geq \frac{1}{\sqrt{\alpha_i}} \frac{\rho}{n}\, \hat{v}_i\Big(\bigcup_{j=1}^{n} A_j^t\Big) \geq \frac{\rho}{n\sqrt{\alpha_i}}\, \hat{v}_i\Big(\bigcup_{j=1}^{n} A_j^t\Big)\,,$$

where again the first and the third inequalities follow from Claim 6.2. Thus, $(A_1^t, \ldots, A_n^t)$ is $\rho/\sqrt{\alpha_i}$-PROP with respect to the original functions $v_1, \ldots, v_n$.

We last argue about MMS. Although the idea is the same, now it is not straightforward to get the last inequality by Claim 6.2 in the chain of inequalities needed. Instead, we are going to relate the maximin shares with respect to the original and to the threshold valuation functions. For a set of goods $S \subseteq M$, let $\boldsymbol{\mu}_i^n(S)$ and $\hat{\boldsymbol{\mu}}_i^n(S)$ be the maximin shares of agent i with respect to v_i and to $\hat{v}_i$, respectively.

Claim 6.3. *For any set of goods $S \subseteq M$ and any agent $i \in N$, it holds that $\boldsymbol{\mu}_i^n(S) \leq \hat{\boldsymbol{\mu}}_i^n(S)$.*

Proof of Claim 6.3. Suppose $\mathcal{T} = (T_1, \ldots, T_n)$ is a maximin share defining partition of S for agent i with respect to v_i, i.e., $\boldsymbol{\mu}_i^n(S) = \min_{T_j \in \mathcal{T}} v_i(T_j)$. Now it is easy to relate the two maximin shares:

$$\hat{\boldsymbol{\mu}}_i^n(S) \geq \min_{T_j \in \mathcal{T}} \hat{v}_i(T_j) \geq \min_{T_j \in \mathcal{T}} v_i(T_j) = \boldsymbol{\mu}_i^n(S)\,,$$

where the first inequality follows by the definition of $\hat{\boldsymbol{\mu}}_i^n(S)$ (that takes the maximum over all such partitions) and the second inequality follows by Claim 6.2. Cl. 6.3 ⊡

Suppose that at the end of some time step t the allocation $(A_1^t, \ldots, A_n^t)$ is ρ-MMS with respect to the threshold functions $\hat{v}_1, \ldots, \hat{v}_n$. Then, for any $i \in N$, we have

$$v_i(A_i^t) \geq \frac{1}{\sqrt{\alpha_i}}\, \hat{v}_i(A_i^t) \geq \frac{1}{\sqrt{\alpha_i}}\, \rho\, \hat{\boldsymbol{\mu}}_i^n\Big(\bigcup_{j=1}^{n} A_j^t\Big) \geq \frac{\rho}{\sqrt{\alpha_i}}\, \boldsymbol{\mu}_i^n\Big(\bigcup_{j=1}^{n} A_j^t\Big)\,,$$

where, as usual the first and third inequalities follow from Claim 6.2. Thus, $(A_1^t, \ldots, A_n^t)$ is $\rho/\sqrt{\alpha_i}$-MMS with respect to the original functions $v_1, \ldots, v_n$. □

For any personalized interval-restricted instance let $\alpha_* := \max_{i \in N} \sqrt{\alpha_i}$. Then Theorem 6.1, combined with Corollary 4.5 or Theorem 5.4, directly implies the following corollaries.

Corollary 6.4. *For any personalized interval-restricted instance, we can construct a $1/\sqrt{a_*}(2n-1)$-temporal-MMS allocation. Moreover, for any agent i who sees at least n high-valued goods with respect to $\hat{v}_i$ of Theorem 6.1, this guarantee eventually improves to $\Omega(1/\sqrt{a_i})$.*

Corollary 6.5. *For any personalized interval-restricted instance augmented with foresight of length $n-1$, we can construct a $1/\sqrt{a_*}$-temporal-EF2 allocation that is also $1/\sqrt{a_*}$-EF1 (and, thus, $1/n\sqrt{a_*}$-MMS) for every time step $t = kn, k \in \mathbb{Z}_{\geq 0}$. If at any step t_0 the allocation fails to be $1/2\sqrt{a_*}$-EF1, then it remains $1/2\sqrt{a_*}$-EF1 at the end of every time step $t \geq \lceil t_0/n \rceil n$.*

7 Discussion and Open Questions

In this paper we study a prior-free online fair division setting where the items are indivisible goods and we focus on the design of deterministic algorithms with solid worst-case fairness guarantees that hold frequently, if not at every time step. By restricting the input space to personalized 2-value (or interval-restricted) instances we are able to obtain nontrivial guarantees that are not possible for the general additive case. We see this as a main take-home message of this work; despite the existence of strong impossibility results, there are meaningful restrictions which can lead to technically interesting findings, broadening our understanding of online fair division. So, the most natural direction for future work is to identify such restrictions and push the boundaries of positive results accordingly.

Another promising direction is to fully explore the power of knowing the future. Our work does not answer whether it is possible to efficiently utilize foresight which is *sublinear* in n for personalized 2-value instances. In fact, Algorithms 2 and 3 only use the (linear) information they have every n steps and ignore it otherwise. We suspect that it is possible to design algorithms that build temporal-EF1 allocations with linear foresight in our setting, although there are simple examples suggesting that this cannot be done via balanced allocations (like the ones our algorithms construct).

Finally, given that personalized interval-restricted instances are already very expressive, it would be particularly interesting to get tight results directly for those. Although the dependency on α_* cannot be completely removed (since for large enough α_* the impossibility results of He et al. [2019] and Zhou et al. [2023] can be replicated), it is likely that the guarantees of Corollaries 6.4 and 6.5 are not tight.

Acknowledgments

This work was partially supported by the project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

This work was partially supported by the framework of the H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union – NextGenerationEU (H.F.R.I. Project Number: 15877).

This work was partially supported by the NWO Veni project No. VI.Veni.192.153.

References

Hannaneh Akrami, Bhaskar Ray Chaudhury, Martin Hoefer, Kurt Mehlhorn, Marco Schmalhofer, Golnoosh Shahkarami, Giovanna Varricchio, Quentin Vermande, and Ernest van Wijland. Maximizing nash social welfare in 2-value instances. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 4760–4767. AAAI Press, 2022.

Martin Aleksandrov and Toby Walsh. Pure nash equilibria in online fair division. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 42–48. ijcai.org, 2017.

Martin Aleksandrov and Toby Walsh. Monotone and online fair division. In *KI 2019: Advances in Artificial Intelligence - 42nd German Conference on AI*, volume 11793 of *Lecture Notes in Computer Science*, pages 60–75. Springer, 2019.

Martin Aleksandrov and Toby Walsh. Online fair division: A survey. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 13557–13562. AAAI Press, 2020.

Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online fair division: Analysing a food bank problem. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2540–2546, 2015.

Georgios Amanatidis, Georgios Birmpas, and Vangelis Markakis. Comparing approximate relaxations of envy-freeness. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 42–48, 2018.

Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, and Alexandros A. Voudouris. Maximum Nash welfare and other stories about EFX. *Theoretical Computer Science*, 863:69–85, 2021.

Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu. Fair division of indivisible goods: Recent progress and open questions. *Artif. Intell.*, 322:103965, 2023.

Georgios Amanatidis, Aris Filos-Ratsikas, and Alkmini Sgouritsa. Pushing the frontier on approximate EFX allocations. In *Proceedings of the 25th ACM Conference on Economics and Computation, EC 2024*, pages 1268–1286. ACM, 2024.

Haris Aziz, Jeremy Lindsay, Angus Ritossa, and Mashbat Suzuki. Fair allocation of two types of chores. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023*, pages 143–151. ACM, 2023.

Eric Balkanski, Vasilis Gkatzelis, Xizhi Tan, and Cherlin Zhu. Online mechanism design with predictions. In *Proceedings of the 25th ACM Conference on Economics and Computation, EC 2024*, page 1184. ACM, 2024.

Siddhartha Banerjee, Vasilis Gkatzelis, Artur Gorokh, and Billy Jin. Online nash social welfare maximization with predictions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 1–19. SIAM, 2022.

Siddhartha Banerjee, Vasilis Gkatzelis, Safwan Hossain, Billy Jin, Evi Micha, and Nisarg Shah. Proportionally fair online allocation of public goods with predictions. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 20–28. ijcai.org, 2023.

Siddhartha Banerjee, Chamsi Hssaine, and Sean R. Sinclair. Online fair allocation of perishable resources. *CoRR*, abs/2406.02402, 2024. doi: 10.48550/ARXIV.2406.02402. URL `https://doi.org/10.48550/arXiv.2406.02402`.

Siddharth Barman, Arindam Khan, and Arnab Maiti. Universal and tight online algorithms for generalized-mean welfare. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 4793–4800. AAAI Press, 2022.

Gerdus Benade, Aleksandr M. Kazachkov, Ariel D. Procaccia, and Christos-Alexandros Psomas. How to make envy vanish over time. In *Proceedings of the 2018 ACM Conference on Economics and Computation (EC)*, pages 593–610, 2018.

Ziyad Benomar and Vianney Perchet. Non-clairvoyant scheduling with partial predictions. In *Forty-first International Conference on Machine Learning, ICML 2024*. OpenReview.net, 2024.

Erik Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

Benjamin Cookson, Soroush Ebadian, and Nisarg Shah. Temporal fair division. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI-25)*, pages 13727–13734. AAAI Press, 2025.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

Edith Elkind, Alexander Lam, Mohamad Latifian, Tzeh Yuan Neoh, and Nicholas Teh. Temporal fair division of indivisible items. *CoRR*, abs/2410.14593, 2024. doi: 10.48550/ARXIV.2410.14593. URL `https://doi.org/10.48550/arXiv.2410.14593`.

Zack Fitzsimmons, Vignesh Viswanathan, and Yair Zick. On the hardness of fair allocation under ternary valuations. *CoRR*, abs/2403.00943, 2024. doi: 10.48550/ARXIV.2403.00943. URL `https://doi.org/10.48550/arXiv.2403.00943`.

Jugal Garg, Aniket Murhekar, and John Qin. Fair and efficient allocations of chores under bivalued preferences. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 5043–5050. AAAI Press, 2022.

Vasilis Gkatzelis, Alexandros Psomas, and Xizhi Tan. Fair and efficient online allocations with normalized valuations. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 5440–5447. AAAI Press, 2021.

Jiafan He, Ariel D. Procaccia, Alexandros Psomas, and David Zeng. Achieving a fairer future by changing the past. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 343–349, 2019.

Thomas Kalinowski, Nina Narodytska, and Toby Walsh. A social welfare optimal sequential allocation procedure. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 227–233. IJCAI/AAAI, 2013.

Ian A. Kash, Ariel D. Procaccia, and Nisarg Shah. No agent left behind: Dynamic fair division of multiple resources. *Journal of Artificial Intelligence Research*, 51:579–603, 2014.

Pooja Kulkarni, Ruta Mehta, and Parnian Shahkar. Online fair division: Towards ex-post constant MMS guarantees. *CoRR*, abs/2503.02088, 2025. doi: 10.48550/ARXIV.2503.02088. URL `https://doi.org/10.48550/arXiv.2503.02088`.

Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*, pages 125–131, 2004.

Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.

Aniket Murhekar and Jugal Garg. On fair and efficient allocations of indivisible goods. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 5595–5602. AAAI Press, 2021.

Ariel D. Procaccia, Ben Schiffer, and Shirley Zhang. Honor among bandits: No-regret learning for online fair division. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

Sean R. Sinclair, Siddhartha Banerjee, and Christina Lee Yu. Sequential fair allocation: Achieving the optimal envy-efficiency tradeoff curve. *CoRR*, abs/2105.05308, 2021. URL `https://arxiv.org/abs/2105.05308`.

Hugo Steinhaus. Sur la division pragmatique. *Econometrica*, 17 (Supplement):315–319, 1949.

Shai Vardi, Alexandros Psomas, and Eric J. Friedman. Dynamic fair resource division. *Math. Oper. Res.*, 47 (2):945–968, 2022.

Hakuei Yamada, Junpei Komiyama, Kenshi Abe, and Atsushi Iwasaki. Learning fair division from bandit feedback. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2024*, volume 238 of *Proceedings of Machine Learning Research*, pages 3106–3114. PMLR, 2024.

David Zeng and Alexandros Psomas. Fairness-efficiency tradeoffs in dynamic fair division. In *Proceedings of the 21st ACM Conference on Economics and Computation (EC)*, pages 911–912, 2020.

Shengwei Zhou, Rufan Bai, and Xiaowei Wu. Multi-agent online scheduling: MMS allocations for indivisible items. In *International Conference on Machine Learning, ICML 2023*, volume 202 of *Proceedings of Machine Learning Research*, pages 42506–42516. PMLR, 2023.