EPFL

Learning to maximize the social welfare from preference feedback

Sycamore Optional Research Project Report
(École Polytechnique Fédérale de Lausanne)

Antoine Bergerault
Supervised by Anna Maddux, Andreas Schlaginhaufen,
Wenjie Xu, Prof. Maryam Kamgarpour

Spring 2024

Abstract

Designing utilities in games so as to maximize social welfare objectives is a central problem in many applications. We study how this can be done when the social welfare is only accessible via preference feedback provided by a central authority. In particular, we present SDO-PBO, a novel algorithm that we analyze for the case of DR-submodular welfare functions, found in practice in allocation or congestion games. We pave the way on how it can be implemented for the cooling of computer clusters or the optimization of bike-sharing systems, before further discussing the limitations of our work and potential future research directions.

Contents

1 Introduction

In numerous modern applications, multi-agent systems are employed to distribute decision-making across various rational agents, each pursuing their own objectives. This decentralization is often chosen either because communication between agents is infeasible or undesired, such as in autonomous vehicle networks, or because it effectively reduces the complexity associated with centralized optimization, as seen in multi-robot systems.

The field of game theory studies strategic decision-making of rational agents, each aiming to maximize their own perceived utility. In mechanism and utility design, a domain of game theory, we are interested in defining the rules of a game such that those rational agents, by optimizing their own utilities, optimize at the same time a so-called social welfare function. While we can influence the utilities of the agents through game design or incentives, the social welfare function is intrinsic to the optimization problem we want to solve and, therefore, is assumed to be fixed.

In this project, we are interested in the setting where the social welfare function is too complex to formalize mathematically and is thus considered to be a black-box function that a central agent would like to maximize. This scenario has been explored by [1], where they can iteratively query the welfare function to learn it using kernel regression, providing a novel approach and theoretical guarantees for this welfare maximization problem. However, in certain real-world scenarios, it may be impossible to measure the exact value of the social welfare function, but we may still be able to collect preference feedback over different outcomes or action profiles $a \in \mathcal{A}$.

Preference-based feedback is motivated by a set of common observations in applications of multi-agent systems. First, many settings involve a social welfare function that results from different constraints or conflicting objectives. In that case, the former is hard to encompass in a closed mathematical form. Second, it has been shown experimentally that humans are better at comparing pairs of solutions than at sensing absolute magnitudes [2]. In particular, the idea behind preference-based feedback is to ask annotators or human experts to compare pairs of actions and decide which ones would be preferred over the other. For instance, in the area of Natural Language Processing, one might align language models by first collecting data from human annotators, before using this data to further improve the models. While it would be hard to give numerical values on sentences to assess their politeness, it is easy for humans to compare two sentences on this criterion. Based on similar motivations, this project aims to leverage preference feedback on the social welfare function, to iteratively design adequate utilities for the agents in the game.

While multiple algorithms have been proposed for bayesian optimization problems with preference feedback [3, 4], their use in decentralized game-theoretic settings such as in [1] is not well understood. This project aims to fill this gap by integrating preference-based feedback within the framework of mechanism design.

Contributions By combining preferential bayesian optimization with mechanism design, this project explores how a central agent can design rules of the game to maximize a social welfare function $S : \mathcal{A} \to \mathbb{R}$ using a history of preference feedback. In this report, we develop and analyze a novel solution for this question. We first extend the work done by [4] to a game-theoretic setting, and then derive SDO-PBO, an algorithm that we prove to be efficient in the optimization of DR-submodular welfare functions.

2 Applications

Different domains might benefit from our research results. Quite naturally, many applications where reinforcement learning from human feedback is used could potentially leverage preference-based mechanism design to improve their results, but others can be formulated in the framework of this work. We detail here two very different applications, which can benefit from this and future research in that direction.

2.1 Data center cooling systems

This application is inspired by recent work done by Deepmind on optimizing data center and commercial cooling systems [5, 6].

Data centers consume vast amounts of energy, a significant portion of which is dedicated to cooling systems that maintain optimal operating temperatures for servers. The challenge lies in optimizing these cooling systems to reduce energy consumption while ensuring user satisfaction as well as remaining within safe temperature limits for the servers.

Decentralized optimization can be employed to optimize such systems with a high number of different independent cooling components divided into multiple computing clusters. Mechanism-design can be used and help optimize pricing, matchings (or resource allocation), and other parameters to maximize overall system efficiency and user satisfaction. The system could adjust cooling strategies through time to learn and achieve energy efficiency while maintaining performance standards (e.g. avoiding network congestion due to underperforming hardware).

The social welfare function in this context can incorporate factors like energy consumption, user satisfaction, fairness, efficiency, system reliability, and environmental impact. Balancing multiple conflicting objectives in real-time makes it suitable for the unknown social welfare assumption.

By learning the preferences directly from the operating company, derived from gathered data of different entities (e.g. server clusters, cooling units and user retention), cooling systems can be managed more efficiently.

2.2 Optimization of bike-sharing systems

Bike-sharing systems, or more generally, shared mobility systems, represent another domain where our research can have significant impact (similar to [1]). Bike-sharing systems involve the distribution and maintenance of bicycles across a network of stations to meet user demand. The primary challenge is to ensure that bicycles are available where and when they are needed, balancing supply and demand efficiently.

Using preference-based optimization, bike-sharing systems can be optimized by learning from user preferences and feedback, data that can be collected through usage of bikes or estimation of unmet demand. Usually, these types of systems are operated by doing overnight relocation of bikes to multiple stations, using different trucks routed to different parts of the operated city. As such, this problem can be viewed as an allocation problem, a standard type of problems in game theory.

In this application, the social welfare function may consider the total satisfaction of all users, taking into account factors such as bike availability, station accessibility, and user convenience. User decisions viewed as preference feedback can be particularly important in these situations because it can help determine diverse individual valuations for different resources, which might not be straightforwardly quantifiable in numerical terms.

3 Background

3.1 Reproducing Kernel Hilbert Space

In regression problems, the goal is to approximate a function[1] $f : \mathcal{X} \to \mathbb{R}$ from a set of feature points $\{x_i : x_i \in \mathcal{X}\}_{i=1}^n$ and their corresponding responses $\{f(x_i)\}_{i=1}^n$. To do so, we might first want to expand our features x_i to some chosen $\phi(x_i) \in \mathcal{X}'$. This transformation allows us to capture more complex relationships within the data, making our estimations more expressive and accurate. In many cases, however, we prefer to introduce the notion of the corresponding kernel function $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$, allowing infinite-dimensional feature expansion ϕ and leveraging the well-known kernel-trick[2].

For a given ϕ and scalar product $\langle \cdot \rangle$, one can define the unique corresponding kernel function k. Conversely, for any given $k : \mathcal{X}' \times \mathcal{X}' \to \mathbb{R}$ satisfying the conditions of a valid kernel function (Mercer's conditions), there exists a unique Hilbert function space $\mathcal{H}_k$ called the Reproducing Kernel Hilbert Space (RKHS) such that:

1. $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{H}_k}$ for some $\phi : \mathcal{X} \to \mathcal{X}'$ and all $x_1, x_2 \in \mathcal{X}$
2. $\phi(x) = k(x, \cdot) = k(\cdot, x)$ for all $x \in \mathcal{X}$
3. $\mathcal{H}_k$ is k-reproducing: $f(x) = \langle f, \phi(x) \rangle_{\mathcal{H}_k}$ for all $f \in \mathcal{H}_k$

Furthermore, any linear combination of the expanded features $f \in \text{span}(\phi(x_1), \ldots, \phi(x_n))$ can be written as a function of the kernel k:

$$f(x) = \sum_{i=1}^{n} \langle \alpha_i \phi(x_i), \phi(x) \rangle_{\mathcal{H}_k} = \sum_{i=1}^{n} \alpha_i k(x, x_i) = \alpha^\top k(x, x_{1:n})$$

Designing a kernel k amounts to designing the covariance matrix $K = \{k(x_i, x_j)\}_{i,j=1}^t$ of any set of feature points $\{\phi(x_i)\}_{i=1}^t$. For this reason, we usually take k to be of one of the following well-behaving forms:

Linear

$$k(x_1, x_2) = x_1^\top x_2$$

Squared Exponential or Gaussian with parameters $\sigma^2, l \in \mathbb{R}_+$

$$k(x_1, x_2) = \sigma^2 \exp\left(-\frac{|x_1 - x_2|^2}{l^2}\right)$$

Matérn with parameters $\rho, \nu \in \mathbb{R}_+$

$$k(x_1, x_2) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x_1 - x_2|}{\rho}\right)^\nu \mathrm{K}_\nu \left(\frac{\sqrt{2\nu}|x_1 - x_2|}{\rho}\right),$$

Where Γ is the gamma function and K_ν a modified Bessel function [8].

RKHS play an important role in the theory of Gaussian Process (GP) Regression, see [9] for a reference book.

[1] We can more generally perform a regression for learning distributions, but this is not of interest for this report.

[2] The feature expansion principle is common to many machine learning algorithms. In many cases, these new features only interact through their dot product $\langle \phi(x_i), \phi(x_j) \rangle$, for some $i, j \in \{1, ..., n\}$, as a measure of their similarity. Introducing the kernel $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$ can make these dot products more efficient to compute and $\mathcal{X}'$ becomes only an implicit feature space which can therefore be infinite dimensional. For more details on kernel methods, see [7].

3.2 Bayesian Optimization in RKHS

The Bayesian Optimization framework is a probabilistic approach to optimization problems, where kernel regression and Gaussian processes (GP) regression are specific tools. We give here only a brief overview of what it tries to do.

Given an unknown function f to optimize, where f lives in an RKHS $\mathcal{H}_k$, this method iteratively measures realisations $y_t = f(x_t) + \epsilon_t$, ϵ_t being sampled from some given distribution, to iteratively learn surrogate functions $\tilde{f}_t$ approximating f. At time t, y_t being sampled from the true function f, we can further improve $\tilde{f}_t$ by applying Bayes formula and obtaining the posterior distribution for the updated $\tilde{f}_{t+1}$.

Choosing x_t for each time step can be done in multiple ways. One common idea is to couple the surrogate with an acquisition function a, which can represent uncertainty along $\tilde{f}$, expected improvement (EI), optimistic values for $\tilde{f}$ (UCB) or any other metric derived from $\tilde{f}$. The next x_t is then chosen by optimizing over the known acquisition a, or based on some random methods linked to a.

At each time step t, the distribution of $\tilde{f}_t$ can be used to compute a maximum likelihood estimator (MLE) and a confidence set for the true f. While confidence sets can be computed in different ways, the MLE function is defined as

$$f_t^{\text{MLE}} = \arg\max_{\tilde{f} \in \mathcal{H}_k} p_{\tilde{f}}^t(y_1, \dots, y_t),$$

where $p_{\tilde{f}}^t(\cdot)$ is the density function of $(\tilde{f}(x_1), \dots, \tilde{f}(x_t))$, known as the likelihood of $\tilde{f}$ when evaluated at the realizations $\{y_\tau\}_{\tau=1}^t$.

These all together form powerful tools for (non-parametric) regression over any non-empty compact set $\mathcal{X}$, tracking both best function estimate and uncertainty. More details about Bayesian Optimization and likelihood ratio confidence sets can be found in [10, 11].

3.3 Multi-arm bandits (MAB)

In multi-arm bandits (MAB) a set of actions $\mathcal{A}$ is available to a given system, which will then learn by interacting within an environment by playing actions. Once an action $a \in \mathcal{A}$ is chosen to be played, the environment returns a reward vector $r \in \mathbb{R}^{|\mathcal{A}|}$. For every turn t, we denote by a_t and $r_t(a_t)$ the action and reward of this turn.

When the system learns from feedback of the form $(a_\tau, r_\tau(a_\tau))$, we say that it learns from bandit feedback. Full-information feedback is an alternative form of feedback specified by $(a_\tau, \{r_\tau(a)\}_{a \in \mathcal{A}})$, where the system can observe the reward of all actions. Such full-information feedback is commonly used for learning in multi-agent games.

Some common algorithms in the MAB litterature are:

- Exp3 with bandit feedback.
- Hedge or multiplicative weights update (MWU) algorithm [12, 13] with full-information feedback.

In a simultaneous multi-agent game, each player's point of view can be seen as that of a multi-armed bandit problem. Agent i chooses an action a_t^i based on previous feedback from its utility functions $u^i : \mathcal{A} \to \mathbb{R}$ with rewards $\{r_\tau(a_\tau^i)\}_{\tau=1}^t = \{u^i(a_\tau^i, a_\tau^{-i})\}_{\tau=1}^t$, while the other players similarly choose their

actions a_t^{-i}. In order to learn the best joint actions (in the sense of an equilibrium, see Section 3.4 for more details) the agents can run bandit algorithms such as the ones mentioned above in parallel.

3.4 Game Theory

Game theory aims to understand the behavior of rational agents in a game setting. A game comprises multiple agents with their action set, and each is driven by a personal utility function they want to maximize. A game is formally defined as follows:

Definition 3.1 (Normal-form representation of a game)**.** Formally, we characterize a game $\mathcal{G}$ by a tuple comprising the number of players N, a set of individual action spaces $\mathcal{A}_i$ for each player i, a set of utility functions $u_i : \mathcal{A}_1 \times \ldots \times \mathcal{A}_N \to \mathbb{R}$ serving the role of payoffs and a social welfare function (or objective) S

$$\mathcal{G} = (N, \{\mathcal{A}_i\}_{i=1}^N, \{u_i\}_{i=1}^N, S)$$

In this section and the rest of the report, unless stated otherwise, we always assume games to be simultaneous games, where all players play their actions at the same time.

As a first step taken to understand the behavior of such agents in game settings, we define different notions of equilibria. Notions explained here are explained in details in [14]. We use the notation $[j] \doteq \{1, \ldots, j\}$, and it will be used throughout this report.

Pure Nash Equilibrium A pure Nash equilibrium (PNE) is an action profile $(a^i)_{i\in[N]} \in \mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_N$, where no player would be better off by unilaterally deviating from its action, i.e.,

$$u_i(\tilde{a}^i, a^{-i}) \leq u_i(a^i, a^{-i}), \quad \forall i \in [N], \forall \tilde{a}^i \in \mathcal{A}_i,$$

where we define $a^{-i} = (a^j)_{j\in[N]\setminus\{i\}}$ as the joint action of all players except i.

Mixed Nash Equilibrium A mixed Nash equilibrium (MNE) is an extension of a Nash equilibrium where players are allowed to employ mixed strategies s_i, which are random variables distributed on $\mathcal{A}_i$ modeling randomization over actions. These constitute a mixed Nash equilibrium when we have

$$\mathbb{E}[u_i(\tilde{s}^i, s^{-i})] \leq \mathbb{E}[u_i(s^i, s^{-i})], \quad \forall i \in [N], \forall \tilde{s}^i \in \Delta\mathcal{A}_i,$$

where $\Delta\mathcal{A}_i$ denotes distributions over $\mathcal{A}_i$, and $s^{-i} = (s^j)_{j\in[N]\setminus\{i\}}$.

Correlated Equilibrium A distribution σ over action profiles $\mathcal{A}$ is a correlated equilibrium (CE) if no player would be better off deviating from this strategy

$$\mathbb{E}_{s\sim\sigma}[u_i(\tilde{s}^i, s^{-i}) \mid s^i] \leq \mathbb{E}_{s\sim\sigma}[u_i(s^i, s^{-i}) \mid s^i], \quad \forall i \in [N], \forall \tilde{s}^i \in \Delta\mathcal{A}_i,$$

Coarse Correlated Equilibrium Similar to a correlated equilibrium, a distribution σ over action profiles $\mathcal{A}$ is a coarse correlated equilibrium (CCE) if no player would be better off choosing another strategy, independently of σ

$$\mathbb{E}_{s\sim\sigma}[u_i(\tilde{s}^i, s^{-i})] \leq \mathbb{E}_{s\sim\sigma}[u_i(s^i, s^{-i})], \quad \forall i \in [N], \forall \tilde{s}^i \in \Delta\mathcal{A}_i,$$

Even though one might want to compute pure Nash equilibria over the other ones, they do not always exist. Even if they exist, computing Nash equilibria is a hard problem with no known polynomial time algorithm in general. In the general case, only CE and CCE are known to be in P.

The following figure (Figure 1) gives a hierarchy of these notions of equilibria. The stronger notions are the hardest to compute, and in practice we usually hope to reach coarse correlated equilibria using equilibrium-finding algorithms, as described in Section 3.3.

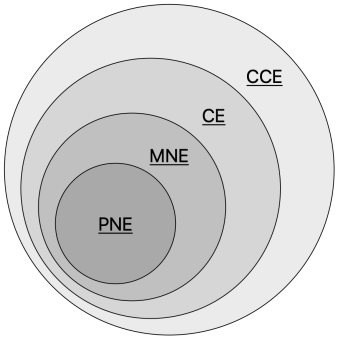

Figure 1: Hierarchy of equilibria types

In particular, computing coarse correlated equilibria can be done in polynomial time with no-regret algorithms. Such algorithms minimize the following (external) regret for every player i:

$$R_{i,T}^{\text{external}} = \max_{a \in \mathcal{A}_i} \sum_{t=1}^{T} u_i(a^i, a_t^{-i}) - \sum_{t=1}^{T} u_i(a_t)$$

As $T \to \infty$, no-regret algorithms experience a time-averaged regret converging to 0. Some examples of such algorithms Exp3 (for bandit feedback) or Hedge (when full feedback is available).

4 Problem Setting

4.1 Overview

We would like to do distributed online maximization of the unknown welfare function S, by formulating the optimization problem as a game with N players. In some settings, this is the natural approach, where communication between different agents of the system is not possible. As the action space grows exponentially with the number of agents, settings where full communication is available might also benefit from this approach, as in this case optimizing the objective centrally is not practically feasible. Combining recent advancements in Bayesian optimization from preferential data with concepts from mechanism design, we propose through this project a novel algorithm and prove its theoretical guarantees for decentrally optimizing DR-submodular objectives from preference feedback.

In previous work [1], a resource allocation problem with r ressources and an unknown submodular welfare function is modeled as a repeated game where multiple rational agents try to maximize their cumulated rewards $\sum_{t=1}^{T} u_t^i(a_t)$ for each agent i. The key idea to make this learning optimal for the welfare function $S(a_t) = \sum_{r=1}^{R} S^r(a_t)$ is to let the central authority design the utility functions u_t^i as a linear combination of time-varying upper confidence bounds on the unknown welfare rewards.[3] More precisely, each component of the unknown welfare function $S^r(\cdot)$ is approximated and tracked via Bayesian Optimization techniques, and the paper reveals that theoretical guarantees can be derived from designing rewards using these approximations.

Our setting goes further by considering that the value of the social welfare $S(a_t)$ is not directly observed. Indeed, we wish not to learn how to correctly design u_t^i from cardinal feedback from the central authority, but rather from ordinal pairwise feedback collected centrally (via for example Human-in-the-loop). We assume that these feedbacks are truthful and are assumed to be coming from a stochastic preference function $\succcurlyeq_P$. Similar to [4], we model the centralized preference feedback by a Bradley-Terry model [15] $1_{a \succcurlyeq_P a'} \sim \text{Ber}(\sigma(S(a) - S(a')))$, where $\text{Ber}(\cdot)$ denotes the Bernoulli distribution and $\sigma(u) = 1/(1+e^{-u})$ is the standard sigmoid function. This choice of preference model is very common in the literature, and in particular gained popularity in reinforcement learning from human feedback [16, 17]. The entire problem setting is summarized by Figure 2.

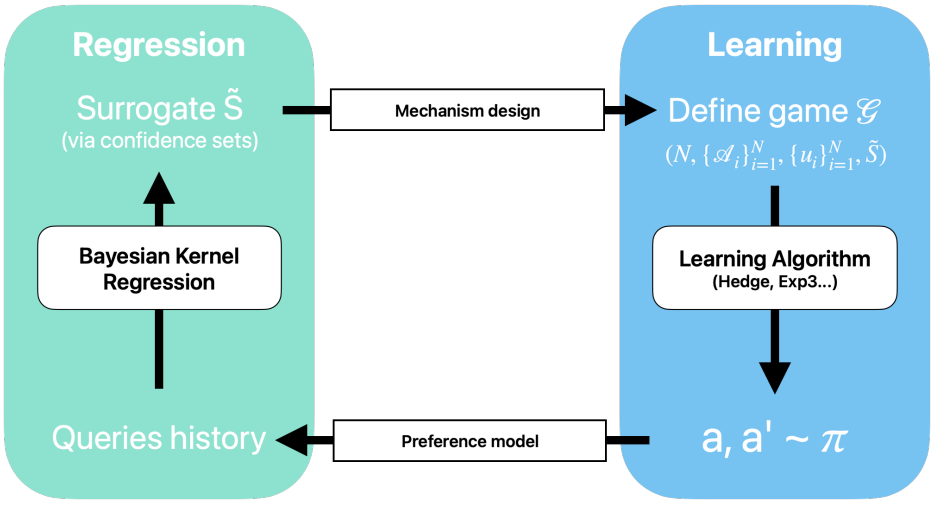

Figure 2: Problem Setting Diagram

[3] They also introduce the idea of a context z_t for each game. As we do not consider such complexifications, we prefer to ignore any mention of such contexts.

In order to be able to learn or approximate the unknown social welfare function S, we further impose the following common assumptions[4]:

Assumption 1. *$\mathcal{A}$ is compact and non-empty.*

Assumption 2. *To learn the social welfare using kernel regression, we first assume that $S \in \mathcal{H}_k$, where $\mathcal{H}_k$ is the RKHS corresponding to a symmetric and positive semidefinite kernel function $k : \mathbb{R}^{Nd} \times \mathbb{R}^{Nd} \to \mathbb{R}$. Moreover, there exists some bounding factor B such that $\|S\|_k \leq B$, where $\|\cdot\|_k$ is the induced-norm of $\mathcal{H}_k$.*

Assumption 3. *The kernel function k is continuous on $\mathbb{R}^{Nd} \times \mathbb{R}^{Nd}$ and satisfies $k(a, a') \leq 1 \ \forall a, a' \in \mathcal{A}$.*

This assumption holds for all the kernels introduced in Section 3.1 after proper normalization.

4.2 Modeling the welfare function

It turns out that the problem can be formulated in two different ways:

1. We model the social welfare function directly, by learning a surrogate $\tilde{S} : \mathcal{A} \to \mathbb{R}$. We then use this surrogate as a tool for designing the utilities of the agents, but it might be that no maximizer a_* of S corresponds to a Nash or Coarse Correlated Equilibrium reached by the agents.
2. Instead of learning the welfare function, we can alternatively learn the utilities directly, incorporating the learning dynamics in the modeling. As such we can optimize over the utilities, taking into account that the learning algorithm might output an equilibrium that is not the welfare function maximizer.

Learning the appropriate utilities $\{u_i\}_{i\in[N]}$ directly is not a trivial task, and is still an open-question on how to do it efficiently in general. For this reason, we chose the first option, where we model S directly, without incorporating the learning dynamics into the regression. See Appendix A.1 for a more detailed discussion.

4.3 Notations

Throughout the report we will additionally use the following notations, that will keep the same meaning in the following sections.

- $\mathcal{A}^i \subseteq [0, a_{\max}]^d$ are individual action spaces, where $d \in \mathbb{N}$ is the dimension of the action space for each player.
- $\mathcal{A} \subseteq \mathbb{R}^{Nd}$ it the joint action space of all the players, defined as $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$
- $u_t^i : \mathcal{A} \to \mathbb{R}$ is the utility function of player i at time t.
- T is the horizon of the game (the number of turns performed).
- $\text{OPT} = S(a_*)$ corresponds to the best achievable social welfare, achieved by playing the actions $a_* \in \arg\max_{a\in\mathcal{A}} S(a)$.
- $\log(\cdot)$ denotes the natural log.

We further introduce the following common abuse of notations:

- When $a \in \mathcal{A}$ represents the joint actions of the players, we decompose a into a^i and a^{-i} for every player i, respectively denoting the individual action of player i and the actions of the other players.
- We use $u_t^i(a)$ and $u_t^i(a^i, a^{-i})$ interchangeably to describe player i's utility when playing $a \in \mathcal{A}$.

[4]These assumptions are mostly taken or adapted from [4]

5 Simplified setting

This project is composed of two parts, one part dedicated to the regression problem, to iteratively learn an approximation of the welfare function from the history of preferences, and one part for the design and optimization of a game played by the agents. As a first step to derive a complete algorithm for this problem, we relax the setting and focus only on the regression part, while ignoring the game-theoretic components. This warmup is an adaptation of [4], and will be the base for the more complete algorithm in Section 6.

In [4], duels[5] are iteratively performed over time with (a_{t-1}, a_t), where $a_t = \max_{a \in \mathcal{A}} s(a)$ maximizes an uncertainty estimate s. This uncertainty estimate s is an optimistic improvement over last actions a_{t-1}, based on the common optimistic design principle for sequential decision making [18, 19]. We draw from this idea to develop the regression part of our problem, that we describe below.

Let $\mathcal{B}_S = \{\tilde{S} \in \mathcal{S} \mid \|\tilde{S}\| \leq B\}$ be the initial set of possible values for the social welfare function S. As new feedbacks are being made available, we iteratively estimate new confidence sets based on the historical data up to time t. Let us first define the likelihood ratio statistic as follows:

Definition 5.1 (Likelihood ratio statistic)**.** We define the likelihood ratio statistic [20] of the function $\tilde{S} \in \mathcal{B}_S$ at time t as a measure of it's likelihood compared to the maximum possible likelihood

$$\Lambda_t(\tilde{S}) = \frac{\mathbb{P}_{\tilde{S}}\left((a_\tau, a_{\tau-1}, 1_{a_\tau \succcurlyeq_P a_{\tau-1}})_{\tau=1}^t\right)}{\mathbb{P}_{S_t^{\text{MLE}}}\left((a_\tau, a_{\tau-1}, 1_{a_\tau \succcurlyeq_P a_{\tau-1}})_{\tau=1}^t\right)},$$

where S_t^{MLE} corresponds to the maximum likelihood estimator of S on the historical data $(a_\tau, a_{\tau-1}, 1_{a_\tau \succcurlyeq_P a_{\tau-1}})_{\tau=1}^t$, where the randomness is taken with respect to the Bradley-Terry preference model for the null and alternative hypotheses $H_0 : S = \tilde{S}$ and $H_1 : S = S_t^{\text{MLE}}$.

Then, we derive confidence sets for every step t based on the likelihood ratio statistic. Using some tolerance factors ϵ, δ and the covering number $\mathcal{N}(\mathcal{B}_S, \epsilon, \|\cdot\|_\infty)$[6] to describe the initial set, we can derive the following theorem.

Theorem 5.1 (Likelihood-based Confidence Set, adapted from Theorem 5 in [4])**.** *The true welfare function S is contained with probability at least $1 - \delta$ in all the following confidence sets:*

$$\mathcal{B}_S^{t+1} = \{\tilde{S} \in \mathcal{B}_S \mid \Lambda_t(\tilde{S}) \geq c_{t,\epsilon,\delta}\},$$

where $c_{t,\epsilon,\delta}$ is such that $\beta_1(\epsilon, \delta, t) = \log(1/c_{t,\epsilon,\delta}) = \mathcal{O}\left(\sqrt{T \log \frac{t\mathcal{N}(\mathcal{B}_S, \epsilon, \|\cdot\|_\infty)}{\delta}} + \epsilon t\right)$.
This corresponds to functions satisfying a likelihood-ratio test with threshold $c_{t,\epsilon,\delta}$. The exact definition of $\beta_1(\epsilon, \delta, t)$ is given in Appendix C.1.

Proof. We refer the reader to the proof of Theorem 5 in [4], based on the theoretical results developed in the same work. □

Using these elements we construct Algorithm 1, an application of POP-BO [4] to multi-agent systems with unknown social welfare function. In classical Bayesian optimization terms, we use an upper confidence bound as the acquisition function [10], depending on a confidence region derived from historical data. We refer the reader to Appendix A.2 for a deeper comparison with previous work.

[5] In this context, a duel means a comparison between two possible joint actions a_1 and a_2.

[6] The covering number $\mathcal{N}(S, \epsilon, \|\cdot\|)$ is the minimum number of balls b of size $\|b\| = \epsilon$ needed to cover the space S, where $\|\cdot\|$ is a norm function

Algorithm 1: Oracle-based decentralized optimistic preferential bayesian optimization

1 Given the initial point $a_0 \in \mathcal{A}$ and set $\mathcal{B}_S^1 = \mathcal{B}_S$
2 **for** $t \in [T]$ **do**
3 Define

$$\overline{\text{adv}}_t(\cdot) = \max_{\tilde{S} \in \mathcal{B}_S^t} \left[\tilde{S}(\cdot) - \tilde{S}(a_{t-1}) \right]$$

as a function of $a \in \mathcal{A}$
4 Set all player utilities to be $u_t(a) = \overline{\text{adv}}_t(a)$ for all $a \in \mathcal{A}$
5 Retrieve actions $a_t = o(\overline{\text{adv}}_t) = \arg\max_{a \in \mathcal{A}} \overline{\text{adv}}_t(a)$
6 Measure preference between a_{t-1} and a_t.
7 Create new confidence set $\mathcal{B}_S^{t+1}$.
8 **end**

To compute the expression in line 3 of the above algorithm, we use the efficient formulation developed in [4] (Theorem 8). In line 5, we denote the maximization problem by the function o. Formally, o is a functional mapping from the function space $\overline{\text{adv}}_t : \mathcal{A} \to \mathbb{R}$ to the action space $\mathcal{A}$. Specifically, to analyse the algorithm and for a practical use we assume the following simplifying condition to hold.

Assumption 4. *There exists a tractable optimization oracle o that can maximize for any time t the function $\overline{\text{adv}}_t$ over the action space $\mathcal{A}$.*

Hence, this simplified setting does not incorporate any learning from the agents. Such a black-box oracle is in general not available for complex objectives requiring distributed optimization, making this setting a special case of preferential Bayesian optimization. When the action space $\mathcal{A} \subseteq \mathbb{R}^{Nd}$ is sufficiently low-dimensional, o can be set to be a grid-search algorithm or a numerical optimizer, and the algorithm can be used for centralized optimization. While such function/algorithm o will not be known in general, good (distributed) optimizers might work as a replacement in practice. Section 6 will provide more theoretical material on this aspect.

During the entire process, we do not distinguish between different agents, and assign them the same utility function at each time step. This corresponds to an identical interest game. This is a choice that will be challenged in Section 6, but ensures for our case that with high probability, a_t converges to a_* at convergence[7].

Intuitively, we progressively refine our initial confidence set $\mathcal{B}_S$ by appending information from the history. $\mathcal{B}_S^t$ becomes smaller and smaller with time, and if the algorithm is consistent, then the optimistic difference will eventually be close to the true difference: $\overline{\text{adv}}_t(\cdot) \approx S(\cdot) - S(a_{t-1})$.

Over all queries history $\omega_{1:T} \in \Omega^T$ with $\Omega \doteq \{(a, a') \mid a, a' \in \mathcal{A}\}$, we further characterize the complexity of learning welfare differences using a kernel-dependent quantity called the maximum information gain [21]. It is a measure of maximum uncertainty reduction at time T over all possible queried pairs. For the clarity of the argument, intuition about this quantity and its role on uncertainty reduction can be found in Appendix A.3.

$$\gamma_T^{SS'} \doteq \max_{\omega_{1:T} \in \Omega^T} \frac{1}{2\lambda} \log \left| K_{t-1}^{SS'} + \lambda I \right|$$

Where $K_{t-1}^{SS'} = \{k(\omega_i) + k(\omega_j)\}_{i,j=1}^{t-1}$

[7] At convergence (with high probability), $\overline{\text{adv}}_t = S(\cdot) - S(a_{t-1})$. Optimizing $\overline{\text{adv}}_t$, we get $a_t = a_*$, a fixed point of the algorithm from there.

With this definition, we are able to prove that the algorithm achieves no-regret with high probability as $T \to \infty$. Formally, the following theorem provides theoretical guarantees for Algorithm 1.

Theorem 5.2 (Adapted from Theorem 9 in [4])**.** *Under the assumptions above, Algorithm 1 achieves with probability at least* $1 - \delta$ *a cumulative regret* R_T *satisfying*

$$R_T = \mathcal{O}\left(\sqrt{\beta_T \gamma_T^{SS'} T}\right),$$

where

$$\beta_T = \beta(1/T, \delta, T) = \mathcal{O}\left(\sqrt{T \log \frac{T\mathcal{N}(\mathcal{B}_S, 1/T, \|\cdot\|_\infty)}{\delta}}\right),$$

$$R_T = \sum_{t=1}^{T} (S(a_*) - S(a_t))$$

with β *as defined in Appendix C.1.*

Proof outline. The result follows from upper bounding the difference in log-likelihood using the Azuma-Hoeffding inequality, combined with a second bound using the covering number $\mathcal{N}(\mathcal{B}_S, 1/T, \|\cdot\|_\infty)$, used to bound any pointwise difference between functions of the ball $\mathcal{B}_S$.

For a detailed proof, we refer the reader to the proof of Theorem 9 in [4], based on the theoretical results developed in the same work. □

This algorithm has been implemented and tested on different test functions using the main framework created along this report. Details of the evaluations can be found in Appendix B.2.1. In Table 1, we evaluate it on different test functions. As an application of POP-BO from [4], we refer the reader to Table 2 in [4] for a comparison with other algorithms.

	Objective	B	Kernel lengthscale	Time to convergence	Highest value achieved
0	bukin	4	4	8	0.99
1	cross in tray	8	4	12	0.39
3	beale	8	4	14	0.99

Table 1: Performance of Algorithn 1 for the best found hyperparameters on common test functions.

6 Optimizing sub-modular welfare functions

We now consider a more realistic setting where we do not have access to an optimization oracle o as defined above. Instead, we model the problem as a repeated game, where we design each players utilities such that they are aligned with the optimistic estimate of the social welfare function of that round. We then iteratively estimate the best actions to take at each turn using equilibrium-finding algorithms.

Furthermore, we assume the true welfare function S to be DR-submodular, a reasonable assumption for many types of games modeling diminishing returns of the different actions in $\mathcal{A}$. For example, this happens naturally in allocation games, where the marginal returns of a position reduces with the number of players or resources already allocated.

Definition 6.1 (DR-submodularity)**.** A function $f : \mathcal{X} \subseteq \mathbb{R}^d \to \mathbb{R}$ is said to be DR-submodular if for every $x \leq y \in \mathcal{X}$, $\forall i \in [d]$, $\forall k \geq 0$ such that $(x + ke_i), (y + ke_i) \in \mathcal{X}$, we have

$$f(x + ke_i) - f(x) \geq f(y + ke_i) - f(y)$$

If f is twice differentiable, this is equivalent to saying that the hessian of f is negative semi-definite.

Assumption 5. *The social welfare function S is DR-submodular. We interpret the actions $\mathbf{0} \in \mathcal{A}$ as not taking any actions, and in such case $S(\mathbf{0}) = 0$.*

This property motivates the study of the marginal contributions (see for example the marginal contribution distribution rule in [22], or the wonderful life utility design [23]). Indeed, DR-submodularity implies that the social welfare is lower bounded by the sum of the marginal contributions of the players. As a consequence, maximizing each agent's marginal contribution independently can help us converge to the maximizer a_*. Connecting this principle with the kernel regression perspective, we define the optimistic marginal contribution as an upper confidence bound on the true marginal contribution of a given player.

Definition 6.2 (Optimistic Marginal Contribution)**.** For player i at time t, we define its optimistic marginal contribution as:

$$\overline{\text{marg}}_t^i(a^i, a^{-i}) = \max_{\tilde{S} \in \mathcal{B}_t} \left[\tilde{S}(a^i, a^{-i}) - \tilde{S}(0, a^{-i}) \right]$$

Over the horizon T, players will have to suffer some cost due to exploration of the different actions and the influence of other players decisions. Given the history of actions $\{a_t\}_{t=1}^T$, we define the learning (external) regret to measure this impact.

Definition 6.3 (Learning regret)**.** For player i, we define its learning (cumulated) regret over the horizon T as the best fixed action it would have chosen in hindsight, knowing the learning dynamics:

$$R^i(T) = \max_a \sum_{t=1}^{T} u_t^i(a, a_t^{-i}) - u_t^i(a_t)$$

An algorithm is said to have the no-regret property if this regret is sub-linear with respect to T. As an example of such algorithms, MWU [13] can be used in the case $\mathcal{A}_i$ is finite for every i, leading to a cumulative regret of order $\mathcal{O}(\sqrt{T \log K})$, where $K = |\mathcal{A}_i|$ is the same for all i.

Let NoRegret represent a no-regret algorithm, used in an adversarial bandit manner where players run parallel instances. During the regression on the social welfare function, we use NoRegret to choose the actions aiming to maximize the optimistic marginal contribution of each player. This is how we construct Algorithm 2, in which we use $\mathcal{M}_t$ to denote the set of actions derived from masking one profile of a_t: $\mathcal{M}_t = \{(a_t^1, \ldots, a_t^{i-1}, 0, a_t^{i+1}, \ldots, a_t^N) \mid i \in [N]\}$.

Algorithm 2: Submodular decentralized optimistic preferential bayesian optimization (SDO-PBO)

1 Initial actions a_0 and set $\mathcal{B}_S^1 = \mathcal{B}_S$
2 Initialize NoRegret algorithm
3 **for** $t \in [T]$ **do**
4 | For every player i, define $u_t^i(a) = \overline{\text{marg}}_t^i(a)$.
5 | Get new actions a_t after one step of NoRegret.
6 | Measure the preference between a_t and $a'_t \in \arg\max_{a' \in \mathcal{M}_t} C_t(a_t, a')$.
7 | Create new confidence set $\mathcal{B}_S^{t+1}$.
8 **end**

Before analyzing the algorithm, we define $C_t(\cdot, \cdot)$ from the following confidence lemma derived from Theorem 7 in [4].

Lemma 6.1 (Confidence lemma)**.** *We have with probability at least $1-\delta$, for all $\hat{S}_t \in \mathcal{B}_S^t, (a, a') \in \mathcal{A} \times \mathcal{A}$,*

$$\left|(\hat{S}_t(a) - \hat{S}_t(a')) - (S_t(a) - S_t(a'))\right| \leq 2C_t(a, a'),$$

where $C_t(a, a') = (2B + \lambda^{-1/2}\sqrt{\beta(\epsilon, \delta/2, t+1)})\sigma_{t+1}^{SS'}((a, a'))$ and $\beta(\epsilon, \delta/2, t) = \mathcal{O}\left(\sqrt{t \log \frac{t\mathcal{N}(\mathcal{B}_S, \epsilon, \|\cdot\|_\infty)}{\delta}}\right)$ (see Appendix C.1 for the definition of β).

Intuitively, $C_t(a, a')$ is a measure of uncertainty on the difference of welfare valuations $S(a) - S(a')$. We use this quantity to select which preference to query on line 6 of Algorithm 2. Specifically, this line measures the preference between a_t and $\arg\max_{a' \in \mathcal{M}_t} C_t(a_t, a')$, corresponding to the action whose preference feedback with a_t is the most uncertain.

Theorem 6.2. *Under the assumptions mentioned above, running Algorithm 2 yields a cumulative welfare $\sum_{t=1}^T S(a_t)$ such that with probability δ we have:*

$$\sum_{t=1}^{T} S(a_t) \geq \alpha \cdot T \cdot OPT - \mathcal{O}\left(N\sqrt{T\beta_T\gamma^{SS'}}\right) - \sum_{i=1}^{N} R_i(T)$$

where $\alpha = (2 - d_{min})^{-1}$, $d_{min} = \inf_i \frac{[\nabla S(2a_{max})]_i}{[\nabla S(0)]_i}$ and the same β_T as defined above and depending on δ.

Proof outline. The algorithm suffers from two aspects:

1. The learning algorithm necessarily introduces sub-optimality and a regret term $R_i(T)$ for each agent i (they all learn independently).
2. The optimistic marginal contribution function is always an upper bound on the true marginal contribution function, which is by itself only a proxy to find the optimal actions a_*. This necessarily worsen the bounds by adding two terms functions of the smoothness of S (the α term) and the complexity of the RKHS regression setup (the $\beta_T\gamma^{SS'}$ term).

Adding these contributions together and upper bounding using the maximal optimistic marginal contribution, we get the desired result. For a more detailed proof, see C.2. □

Theorem 6.2 is quite general and works for all the kernels satisfying the assumptions defined in Section 4.1. In practice, we usually use one of the three mainly used kernels: linear, exponential or Matérn. We present below the corresponding bounds for these kernels (derived from Theorem 11 in [4], using kernel information gains from Theorem 5 in [21]), when Hedge is used as the No-Regret algorithm. We additionally assume that each player has a finite number of actions $K = |\mathcal{A}_i|$, and it is the same for every player i.

Corollary 6.2.1 (Linear kernel)**.** *If we use a linear kernel*

$$\sum_{t=1}^{T} S(a_t) \geq \alpha \cdot T \cdot OPT - \mathcal{O}\left(NT^{3/4}(\log T)^{3/4}\right) - \mathcal{O}\left(N\sqrt{T \log K}\right)$$

Corollary 6.2.2 (Exponential kernel)**.** *If we use an exponential kernel with any parameters*

$$\sum_{t=1}^{T} S(a_t) \geq \alpha \cdot T \cdot OPT - \mathcal{O}\left(NT^{3/4}(\log T)^{3/4(d+1)}\right) - \mathcal{O}\left(N\sqrt{T \log K}\right)$$

Corollary 6.2.3 (Matérn kernel)**.** *If we use a Matérn kernel with parameters (ρ, ν), we get*

$$\sum_{t=1}^{T} S(a_t) \geq \alpha \cdot T \cdot OPT - \mathcal{O}\left(NT^{3/4}(\log T)^{3/4} T^{\frac{Nd}{\nu}\left(\frac{1}{4} + \frac{Nd+1}{4+2(Nd+1)Nd/\nu}\right)}\right) - \mathcal{O}\left(N\sqrt{T \log K}\right)$$

when $\nu > \frac{Nd}{4}(3 + Nd + \sqrt{N^2d^2 + 14Nd + 17}) = \Theta(N^2d^2)$

For the implementation of the algorithm, evaluating the optimistic marginal contribution function is not straightforward. However, it is possible to rewrite it in an efficient formulation that can be solved using common convex optimization solvers. Indeed, the particular form of the problem allows an easier formulation using the representer's theorem. Information about this efficient formulation and the implementation can be found in Appendix C.3.

Although no analysis of game specific metrics or complex experiments involving more than two players and 20 actions were performed, we still ran our implementation on the same test functions used before. As the DR-submodularity property does not hold for these functions we only provide results for reference and describe how to reproduce them in Appendix B.2.2.

7 Conclusion

In this project, we studied the use of preferential bayesian optimization algorithms in game settings. Using an optimistic approach based on marginal contributions of the players, we showed how this method can be applied to the optimization of DR-submodular objectives, where only preference feedback over the social welfare function is available. We implemented and open-sourced the ideas presented here, with the goal of serving as a toolbox for the analysis and integration of PBO algorithms in game settings.

Future work Continuing in that direction, different work could be done to improve the current results. On the theory side, theoretical analysis are still needed when considering different assumptions, as we do not know about any existing literature on mechanism design from preference feedback. For a more practical aspect, implementation in real-world settings still have to be made, demonstrating the strength and limits of the derived algorithms for solving optimization problems.

References

1. Sessa, P. G., Bogunovic, I., Krause, A. & Kamgarpour, M. *Online submodular resource allocation with applications to rebalancing shared mobility systems* in *International Conference on Machine Learning* (2021), 9455–9464.
2. Kahneman, D. & Tversky, A. in *Handbook of the fundamentals of financial decision making: Part I* 99–127 (World Scientific, 2013).
3. González, J., Dai, Z., Damianou, A. & Lawrence, N. D. *Preferential bayesian optimization* in *International Conference on Machine Learning* (2017), 1282–1291.
4. Xu, W., Wang, W., Jiang, Y., Svetozarevic, B. & Jones, C. N. Principled Preferential Bayesian Optimization. *arXiv preprint arXiv:2402.05367* (2024).
5. Gamble, C. & Gao, J. *Safety-first ai for autonomous data centre cooling and industrial control.* `https://deepmind.google/discover/blog/safety-first-ai-for-autonomous-data-centre-cooling-and-industrial-control/`. Accessed: 2024-05-25. 2018.
6. Luo, J. *et al.* Controlling commercial cooling systems using reinforcement learning. *arXiv preprint arXiv:2211.07357* (2022).
7. Hofmann, T., Schölkopf, B. & Smola, A. J. Kernel methods in machine learning (2008).
8. Abramowitz, M. & Stegun, I. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables* ISBN: 9780486612720. `https://books.google.ch/books?id=MtU8uP7XMvoC` (Dover Publications, 1965).
9. Rasmussen, C. E. & Williams, C. K. I. *Gaussian Processes for Machine Learning* (MIT Press, 2006).
10. Frazier, P. I. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).
11. Emmenegger, N., Mutny, M. & Krause, A. Likelihood ratio confidence sets for sequential decision making. *Advances in Neural Information Processing Systems* **36** (2024).
12. Littlestone, N. & Warmuth, M. K. The weighted majority algorithm. *Information and computation* **108,** 212–261 (1994).
13. Freund, Y. & Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* **55,** 119–139 (1997).
14. Roughgarden, T. *Twenty lectures on algorithmic game theory* (Cambridge University Press, 2016).
15. Bradley, R. A. & Terry, M. E. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* **39,** 324–345 (1952).
16. Christiano, P. F. *et al.* Deep reinforcement learning from human preferences. *Advances in neural information processing systems* **30** (2017).
17. Rafailov, R. *et al.* Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* **36** (2024).
18. Liu, Q., Netrapalli, P., Szepesvari, C. & Jin, C. *Optimistic mle: A generic model-based algorithm for partially observable sequential decision making* in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing* (2023), 363–376.
19. Pacchiano, A., Ball, P., Parker-Holder, J., Choromanski, K. & Roberts, S. *Towards tractable optimism in model-based reinforcement learning* in *Uncertainty in Artificial Intelligence* (2021), 1413–1423.
20. Held, L. & Sabanés Bové, D. Applied statistical inference. *Springer, Berlin Heidelberg, doi* **10,** 16 (2014).
21. Srinivas, N., Krause, A., Kakade, S. M. & Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995* (2009).

22. Marden, J. R. & Wierman, A. Distributed welfare games. *Operations Research* **61,** 155–168 (2013).

23. Wolpert, D. H. & Tumer, K. An introduction to collective intelligence. *arXiv preprint cs/9908014* (1999).

24. Roughgarden, T. Intrinsic robustness of the price of anarchy. *Journal of the ACM (JACM)* **62,** 1–42 (2015).

25. Chandan, R., Paccagnan, D. & Marden, J. R. *When smoothness is not enough: Toward exact quantification and optimization of the price-of-anarchy* in *2019 IEEE 58th Conference on Decision and Control (CDC)* (2019), 4041–4046.

26. Paccagnan, D., Chandan, R. & Marden, J. R. Utility design for distributed resource allocation—part i: Characterizing and optimizing the exact price of anarchy. *IEEE Transactions on Automatic Control* **65,** 4616–4631 (2019).

27. Shapley, L. S. *Notes on the N-Person Game mdash; II: The Value of an N-Person Game* (RAND Corporation, Santa Monica, CA, 1951).

28. Auer, P. *Using upper confidence bounds for online learning* in *Proceedings 41st annual symposium on foundations of computer science* (2000), 270–279.

29. Chowdhury, S. R. & Gopalan, A. *On kernelized multi-armed bandits* in *International Conference on Machine Learning* (2017), 844–853.

30. Sessa, P. G., Kamgarpour, M. & Krause, A. *Bounding inefficiency of equilibria in continuous actions games using submodularity and curvature* in *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), 2017–2027.

31. Schölkopf, B., Herbrich, R. & Smola, A. J. *A generalized representer theorem* in *International conference on computational learning theory* (2001), 416–426.

32. Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B. & Diehl, M. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* **11,** 1–36 (2019).

A Additional remarks

A.1 What mapping should we learn?

Before starting this discussion, we introduce some new notations and definitions to describe our game with normal form $\mathcal{G} = (N, \mathcal{A}, u, S)$, where $u \doteq \{u_i\}_{i=1}^{N}$, associated with the objective function S.

Notation We use $\pi_{\text{NE}}(u)$ to denote the set of Nash Equilibria for a game defined with joint utilities u.

Definition A.1 (Price of Anarchy)**.** Let $\mathcal{G} = (N, \mathcal{A}, u, S)$ be a game associated with the social welfare function S. Then, the price of anarchy is defined as the ratio of the worst Nash equilibrium over the best attainable social welfare,

$$\text{PoA} = \frac{\min_{a \in \pi_{\text{NE}}(u)} S(a)}{\max_{a \in \mathcal{A}} S(a)} \in [0, 1]$$

Definition A.2 (Price of Stability)**.** Let $\mathcal{G} = (N, \mathcal{A}, u, S)$ be a game associated with the social welfare function S. Then, the price of stability is defined as the ratio of the best Nash equilibrium over the best attainable social welfare,

$$\text{PoS} = \frac{\max_{a \in \pi_{\text{NE}}(u)} S(a)}{\max_{a \in \mathcal{A}} S(a)} \in [0, 1]$$

Multiple game-theoretic frameworks have been developed to characterize the price of anarchy, but getting tight bounds remain challenging in general. Even in smooth and generalized-smooth games [24, 25], PoA bounds can be somewhat tight but finding the best (λ, μ) parameters for a game can still be difficult in practice. Some other work has been done outside the (generalized) smoothness framework [26], but constructing utility functions around this theory is not straightforward and we did not consider it for our final solution.

We can, however, discuss the implications of two approaches for estimating the welfare functions via bayesian regression: (1) directly modelling the welfare function by a surrogate $\tilde{S}$, or (2) modelling a mapping from the utilities to the welfare function at equilibrium, effectively learning the mechanisms together with S.

Lemma A.1 (Decentralized game theoretic learning inequality)**.** *For any set of possible utilities $\mathcal{U}$, maximizing the social welfare function in the game-theoretic view aiming for a Nash Equilibrium,*

$$\max_{a \in \mathcal{A}} S(a) \geq \max_{u \in \mathcal{U}} \max_{a \in \pi_{NE}(u)} S(a),$$

with equality when $PoS = 1$.

Lemma A.2 (Decentralized game theoretic learning equality)**.** *The previous lemma can be described with equality using the price of stability:*

$$PoS \times \max_{a \in \mathcal{A}} S(a) = \max_{u \in \mathcal{U}} \max_{a \in \pi_{NE}(u)} S(a)$$

These lemmas explain that if we only model S, then if agents seek a game equilibrium (Nash Equilibrium or Coarse Correlated Equilibrium in practice), the corresponding actions will not be optimal. The price of stability and price of anarchy helps understanding this suboptimality.

So using approach (2) which learns the mechanism directly might allow to cover more space of utilities. Maximizing over this bigger space might lead to utilities ensuring a small price of anarchy. In practice, however, one must admit some assumptions about the structure of this space, in order to optimize over it in the implementation. Here are different ways of doing it:

1. Game-specific design: tailor the structure to the type of the game. Easier or more natural structure for allocation games for example
2. Only learn sampled version of utilities (then use a discrete to continuous converter in games), neglecting high frequencies (smoothness assumptions). Maybe need some Lipshitz assumptions about the the social welfare, and probably some game smoothness analysis for deriving theoretical results.
3. Assume a model/distribution for the utilities. For instance, use some sort of cubic interpolation, beta distribution...

On the other side, we can also explicitly design the utilities as a function of the social welfare [1]. Some theory is developed for different game designs and the have been shown to work in practice.

To summarize the remarks and points for deciding to choose the first approach, here is a quick comparison when deciding to use the model F:

1. $F : \mathcal{A} \to \mathbb{R}$, the social welfare $a \mapsto s$.
   - $F(a)$ sometimes never achieved by rational players, due to the price of anarchy and the price of stability.
   - Choice to make for the utilities: total welfare (TW) design, equal share (ES) design, Shapley value [27] (as a first approximation/idea, $u(a) \approx S(a)$)
   - Procedure (idea): set $u \approx F$. Then query $F(\pi_{\mathrm{NE}}(u)) \in [\mathrm{PoA} \times \max_{a\in\mathcal{A}} S(a), \mathrm{PoS} \times \max_{a\in\mathcal{A}} S(a)]$
   - Easier to model
2. $F : \mathcal{U} \to \mathbb{R}$, the mapping $u \mapsto S(\pi_{\mathrm{NE}}(u))$.
   - Mechanism design: choose $u = \arg\max_{u\in\mathcal{U}} F(u)$.
   - Fixing $u = S$ we get $F(S) = S(\pi_{\mathrm{NE}}(S)) \approx S(\arg\max_{a\in\mathcal{A}} S(a)) = \max_{a\in\mathcal{A}} S(a)$
   - $F(S) \in [\mathrm{PoA} \times \max_{a\in\mathcal{A}} S(a), \mathrm{PoS} \times \max_{a\in\mathcal{A}} S(a)]$
   - $F(u)$ always makes sense.
   - We note that $\mathcal{U}$ might be of very high dimension in order to be useful. Due to the curse of dimensionality, this might not be feasible for the regression.
   - Question: can we really do better than $u(a) \approx S(a)$? (i.e. beat Shapley value)

If we can do better than hand-crafted utilities (ES, TW...), then we might want to use the second method. We choose the first approach as the second one is a much harder problem, as designing the best utility functions for any problem still remains an open question. Furthermore, it would be harder to implement and also need a set of assumptions on the structure of the utilities.

A.2 Comparison with previous algorithms

In Algorithm 3, we show how our first algorithm was derived and how it compares to some of previous work, in particular [1, 4].

Algorithm 3: Comparison of Algorithm 1 with previous work

Set initial parameters and estimates

for $t \in [T]$ **do**

Finding next point

Similar in spirit to upper confidence bounds bandit algorithms [28]. Take action with maximal optimistic outcome.

POP-BO (from [4])

Compute

$$a_t \in \arg \underbrace{\max_{a \in \mathcal{A}}}_{\text{Classical numerical optimizer}} \underbrace{\max_{\tilde{S} \in \mathcal{B}_S^t} (\tilde{S}(a) - \tilde{S}(a_t'))}_{\text{Solved using representer's theorem}}$$

Ours

Define

$$\overline{\text{adv}}_t = \max_{\tilde{S} \in \mathcal{B}_S^t} \left[\tilde{S}(\cdot) - \tilde{S}(a_{t-1}) \right]$$

as a function of $a \in \mathcal{A}$

Design joint utilities u_t from $\overline{\text{adv}}_t$

$\implies$ [1] does it by defining rewards as functions of an upper confidence bound.

$\implies$ The reward for a can be computed online via an optimizer to find $\overline{\text{adv}}_t(a)$.

Getting preference feedback

POP-BO (from [4])

Query the comparison oracle to get the feedback result 1_t and append the new data to $\mathcal{D}_t$.

Ours

Get actions $a_t = o(\overline{\text{adv}}_t)$

$\implies$ At this step, [1] applies a no-regret algorithm.

$\implies$ [1] uses known regret bounds with the DR-submodularity assumption.

Measure the preference $1_{a_{t-1} \succcurlyeq_P a_t} \sim \text{Ber}(\sigma(S(a_{t-1}) - S(a_t)))$ and append the new data to the history $\mathcal{D}_t$.

Updating estimates

Similar

end

A.3 Intuition for the maximum information gain

In Algorithm 1, we iteratively perform queries to measure preference feedback on pairs of actions. The feedback being given sampling from a Bernoulli distribution with logit parameter the true difference in welfare values, we introduce tools to understand the modelisation of this difference.

For each new pair (a_{t-1}, a_t), we denote by $\mathcal{B}_{SS'}$ the confidence set associated to plausible values of the true difference $S(a_t) - S(a_{t-1})$, corresponding to a ball in a new RKHS with the augmented kernel $k^{SS'}((x_1, x_1'), (x_2, x_2')) = k(x_1, x_1') + k(x_2, x_2')$:

$$\mathcal{B}_{SS'} \doteq \{S(a_t, a_{t-1}) = \tilde{S}(a_t) + \tilde{S}'(a_{t-1}) \mid \tilde{S}, \tilde{S}' \in \mathcal{B}_S\}$$

Similar to the posterior standard deviation in classical GP-regression [10], we combine each new pair $\omega = (a_t, a_{t-1})$ with an uncertainty associated to $S(a_t) - S(a_{t-1})$:

$$\left(\sigma_t^{SS'}(\omega)\right)^2 \doteq k^{SS'}(\omega, \omega) - k^{SS'}(\omega_{1:t-1}, \omega)^\top (K_{t-1}^{SS'} + \lambda I)^{-1} k^{SS'}(\omega_{1:t-1}, \omega),$$

for some $\lambda > 0$, using $K_{t-1}^{SS'} = \{k^{SS'}(\omega_i, \omega_j)\}_{i,j=1}^{t-1}$.

Over all queries history $\omega_{1:T}$, we further characterize the complexity of this new set $\mathcal{B}_{SS'}$ using the so-called maximum information gain [21].

$$\gamma_T^{SS'} \doteq \max_{\omega_{1:T}} \frac{1}{2\lambda} \log \left| K_{t-1}^{SS'} + \lambda I \right|$$

This quantity can be used to bound the uncertainty $\sigma_t^{SS'}(\omega)$ over the horizon T, with any arbitrary queries history $\omega_{1:T}$. Similarly to the proof of Lemma 4 in [29],

$$\begin{aligned}
\sum_{t=1}^{T} \sigma_t^{SS'}(\omega_t) &\leq \sqrt{T \sum_{t=1}^{T} \left[\sigma_t^{SS'}(\omega_t)\right]^2} \\
&\leq \sqrt{T\lambda \sum_{t=1}^{T} \lambda^{-1} \left[\sigma_t^{SS'}(\omega)\right]^2} \\
&\leq \sqrt{2T\lambda \sum_{t=1}^{T} \log\left(1 + \lambda^{-1} \left[\sigma_t^{SS'}(\omega)\right]^2\right)} \\
&\leq \sqrt{4T\lambda \sum_{t=1}^{T} \frac{1}{2} \log\left(1 + \left[\sigma_t^{SS'}(\omega)\right]^2\right)} \\
&\leq \sqrt{4T\lambda\gamma_T^{SS'}}
\end{aligned}$$

With $\lambda = 1 + \frac{2}{T}$, we finally get the result of Lemma 4 in [29]. For the last step, we refer the reader to [29] (in particular Lemma 3), while for the other steps we used in order, Cauchy-Schwartz and the inequality $\log(1 + \alpha) \geq \alpha/2 \ \forall \alpha \in [0, 1]$.

B Experiments details

B.1 Framework overview

The experiments in this report were performed using a dedicated framework open-sourced at the following location: https://github.com/Antoine-Bergerault/preference-based-social-welfare-optimization/. We note that some parts of the code are inspired from or leverage several other projects. We refer the reader to the appropriate credits and dependencies present on the GitHub repository.

Using this framework, we can run the approaches presented in this report on multiple welfare functions. This is all integrated in the form of a Python library, that has to be installed for running the project (see the setup). This allows the use of this work in other applications, where it would also be easy to change the computation of the preference feedback by a request for a human response or an experience to be run.

Analyzing the programs can be made through different scripts. These were in part used for generating visuals and experimental results.

B.2 Experiments

B.2.1 Simplified setting

Algorithm 1: Oracle-based decentralized optimistic preferential bayesian optimization

Given the initial point $a_0 \in \mathcal{A}$ and set $\mathcal{B}_S^1 = \mathcal{B}_S$

for $t \in [T]$ **do**

Define

$$\overline{\text{adv}}_t(\cdot) = \max_{\tilde{S} \in \mathcal{B}_S^t} \left[\tilde{S}(\cdot) - \tilde{S}(a_{t-1}) \right]$$

as a function of $a \in \mathcal{A}$

Set all player utilities to be $u_t(a) = \overline{\text{adv}}_t(a)$ for all $a \in \mathcal{A}$

Retrieve actions $a_t = o(\overline{\text{adv}}_t) = \arg\max_{a \in \mathcal{A}} \overline{\text{adv}}_t(a)$

Measure preference between a_{t-1} and a_t.

Create new confidence set $\mathcal{B}_S^{t+1}$.

end

The base structure of this algorithm is the core around which the framework has been developed.

Common optimization functions

Bukin we consider $x_1, x_2 \in [-12, 12]$

$$f(x_1, x_2) = 100\sqrt{|x_2 - 0.01x_1^2|} + 0.01|x_1 + 10|$$

Cross in tray we consider $x_1, x_2 \in [-12, 12]$

$$f(x_1, x_2) = -0.0001 \left[\left| \sin(x_1)\sin(x_2)\exp\left(\left| 100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \right| \right) \right| + 1 \right]^{0.1}$$

Beale we consider $x_1, x_2 \in [-2.4, 2.4]$

$$f(x_1, x_2) = -(1.5 - x_1 + x_1 x_2)^2 - (2.25 - x_1 + x_1 x_2^2)^2 - (2.625 - x_1 + x_1 x_2^3)^2$$

Note the negative signs, as we inverted the traditional beale function

Configurations for the results on the test functions can be loaded from `configs/experiments/experiment-1.yaml`. We further note that we use an additional β hyperparameter and set $\beta_1(\epsilon, \delta, t) = \beta \cdot \sqrt{t+1}$. This simplifies the practical implementation. For all the experiments on these test functions, we use $\beta = 0.01$, a value we determined by hyperparameter search. Additionally, we always use a Gaussian kernels of specified lengthscale. In the report of the results, we normalize the value achieved by our algorithms between 0 and 1, corresponding to the minimum and maximum values in the given function range.

B.2.2 No-regret PBO

As part of this work, we do not provide any experiments for this algorithm. We foresee that this can be done in future work, for both test functions and real-world problems. However, we built the basic tools for using Algorithm 2 (restated below) in our framework.

In the global configuration of the framework, we can set `marginal_ucb=True` in order to use our marginals idea. Similar to the implementation of the simple formulation of Algorithm 1, we use an efficient implementation of the optimistic marginal contribution as described in Appendix C.3.

Algorithm 2: Submodular decentralized optimistic preferential bayesian optimization (SDO-PBO)

Initial actions a_0 and set $\mathcal{B}_S^1 = \mathcal{B}_S$
Initialize NoRegret algorithm
for $t \in [T]$ **do**
  For every player i, define $u_t^i(a) = \overline{\text{marg}}_t^i(a)$.
  Get new actions a_t after one step of NoRegret.
  Measure the preference between a_t and $a_t' \in \arg\max_{a' \in \mathcal{M}_t} C_t(a_t, a')$.
  Create new confidence set $\mathcal{B}_S^{t+1}$.
end

For the no-regret learning algorithms, we only implemented Hedge [13], refered with the name `hedge`, `mw` or `multiplicative_weights` in the framework configuration. Using the same test functions as described above we get the results shown in Table 2. Note that these functions are not DR-submodular, results are presented for reference only. Configurations for the results on the test functions can be loaded from `configs/experiments/experiment-2.yaml`.

	Objective	B	Kernel lengthscale	Time to convergence	Highest value achieved
0	bukin	4	4	14	0.73
1	cross in tray	8	4	14	0.49
3	beale	8	4	15	1.0

Table 2: Performance of SDO-PBO for the best found hyperparameters on common test functions. Hedge is used as the No-Regret algorithm.

C Additional proofs

C.1 Additional notations

Many theoretical results use common notations that were not introduced in the main content for clarity. We mention them here, taking the same meaning as in [4].

Factor on the log-likelihood difference in an ϵ-covering space (with respect to the infinity norm)

$$C_L = 1 + \frac{2}{1 + e^{-2B}}$$

The log-likelihood difference threshold

$$\beta_1(\epsilon, \delta, t) = \sqrt{32tB^2 \log \frac{\pi^2 t^2 \mathcal{N}(\mathcal{B}_S, \epsilon, \|\cdot\|_\infty)}{6\delta}} + C_L \epsilon t$$

Induced additional term for elliptical bound on differences

$$\beta_2(\epsilon, \delta, t) = \frac{(1 + e^{-2B})^2}{4} \epsilon^2 t + 2C_L \epsilon t + \sqrt{8t \left(\frac{1 + e^{2B}}{1 + e^{-2B}} - \frac{1 + e^{-2B}}{1 + e^{2B}} \right)^2 \log \frac{\pi^2 t^2 \mathcal{N}(\mathcal{B}_S, \epsilon, \|\cdot\|_\infty)}{3\delta}}$$

Elliptical bound on differences

$$\beta(\epsilon, \delta, t) = \frac{(1 + e^{-B})^2}{2(e^B + e^{-B})^4} \left(\beta_2(\epsilon, \delta, t) + 2\beta_1(\epsilon, \delta, t) \right)$$

C.2 Proof of Theorem 6.2

Theorem 6.2. *Under the assumptions mentioned above, running Algorithm 2 yields a cumulative welfare* $\sum_{t=1}^{T} S(a_t)$ *such that with probability* δ *we have:*

$$\sum_{t=1}^{T} S(a_t) \geq \alpha \cdot T \cdot OPT - \mathcal{O}\left(N \sqrt{T \beta_T \gamma^{SS'}} \right) - \sum_{i=1}^{N} R_i(T)$$

where $\alpha = (2 - d_{min})^{-1}$, $d_{min} = \inf_i \frac{[\nabla S(2a_{max})]_i}{[\nabla S(0)]_i}$ *and the same* β_T *as defined above and depending on* δ.

Proof. The proof of this theorem is very similar to the proof of Theorem 1 in [1]. For the simplicity of notation we denote

$$C_t \doteq \max_{a' \in \mathcal{M}_t} C_t(a_t, a')$$

$$a_*^{1:i} \doteq (a_*^1, \ldots, a_*^i, 0, \ldots, 0)$$

Where $\mathcal{M}_t = \{(a_t^1, \ldots, a_t^{i-1}, 0, a_t^{i+1}, \ldots, a_t^N) \mid i \in [N]\}$.

$$
\begin{aligned}
\sum_{t=1}^{T} S(a_t) &\overset{1}{\geq} \sum_{t=1}^{T}\sum_{i=1}^{N} S(a_t) - S(0, a_t^{-i}) \\
&\overset{2}{\geq} \sum_{t=1}^{T}\sum_{i=1}^{N} \overline{\text{marg}}_t^i(a_t^i, a_t^{-i}) - N\sum_{t=1}^{T} 2C_t \\
&\overset{3}{\geq} \sum_{t=1}^{T}\sum_{i=1}^{N} \overline{\text{marg}}_t^i(a_*^i, a_t^{-i}) - N\sum_{t=1}^{T} 2C_t - \sum_{i=1}^{N} R^i(T) \\
&\overset{4}{\geq} \sum_{t=1}^{T}\sum_{i=1}^{N} S(a_*^i, a_t^{-i}) - S(0, a_t^{-i}) - N\sum_{t=1}^{T} 2C_t - \sum_{i=1}^{N} R^i(T) \\
&\overset{5}{\geq} \sum_{t=1}^{T}\sum_{i=1}^{N} S(a_t + a_*^{1:i}) - S(a_t + a_*^{1:i-1}) - N\sum_{t=1}^{T} 2C_t - \sum_{i=1}^{N} R^i(T) \\
&\overset{6}{=} \sum_{t=1}^{T} S(a_* + x_t) - S(a_t) - N\sum_{t=1}^{T} 2C_t - \sum_{i=1}^{N} R^i(T) \\
&\overset{7}{\geq} \alpha \cdot T \cdot \text{OPT} - N\sum_{t=1}^{T} 2C_t - \sum_{i=1}^{N} R^i(T) \\
&\overset{8}{\geq} \alpha \cdot T \cdot \text{OPT} - \mathcal{O}\left(N\sqrt{\beta_T T \gamma^{SS'}}\right) - \sum_{i=1}^{N} R^i(T)
\end{aligned}
$$

with $\alpha = (2 - d_{\min})^{-1}$, $d_{\min} = \inf_i \frac{[\nabla S(2a_{max})]_i}{[\nabla S(0)]_i}$.

We explain the steps as follows

1. By DR-submodularity, the sum of the marginals of $S(a_t)$ is always smaller than $S(a_t)$ itself.

2. It is a direct application of Lemma 6.1, where we additionally upper bound every individual $C_t(a_t, (a_t^1, \ldots, a_t^{i-1}, 0, a_t^{i+1}, \ldots, a^N))$ by C_t.

3. By the definition of $R^i(T) = \max_a \sum_{t=1}^{T} \overline{\text{marg}}_t^i(a, a_t^{-i}) - \overline{\text{marg}}_t^i(a_t)$, we get for each player $i \in [N]$

$$
\sum_{t=1}^{T} \overline{\text{marg}}_t^i(a_t^i, a_t^{-i}) = \max_a \sum_{t=1}^{T} \overline{\text{marg}}_t^i(a, a_t^{-i}) - R^i(T) \geq \sum_{t=1}^{T} \overline{\text{marg}}_t^i(a_*^i, a_t^{-i}) - R^i(T)
$$

   The transition from the previous step is done with this inequality summed over the N players.

4. We use at this step the definition of the optimistic marginal contribution on the confidence set $\mathcal{B}_t$

$$
\overline{\text{marg}}_t^i(a^i, a^{-i}) = \max_{\tilde{S} \in \mathcal{B}_t} \left[\tilde{S}(a^i, a^{-i}) - \tilde{S}(0, a^{-i})\right] \geq S(a^i, a^{-i}) - S(0, a^{-i})
$$

   This step is only guaranteed to hold with probability δ.

5. The DR-submodularity property is applied on every term indexed by i, t, specifically

$$S(x_1) - S(x_2) \geq S(x_1 + b) - S(x_2 + b)$$

with $x_1 = (a_*^i, a_t^{-i})$, $x_2 = (0, a_t^{-i})$, $b = a_t^i e_i + a_*^{t-1}$. Using the usual abuse of notation $(a_1^i, a_2^{-i}) = a_1^i e_i + a_2(1 - e_i)$.

6. We simplify the telescopic sum of the previous step

7. We use here a similar reasoning as in the proof of Theorem 1 in [1]

We want to show the following

$$\sum_{t=1}^{T} S(a_* + a_t) - S(a_t) \geq (2 + d_{\min})^{-1} \cdot T \cdot \mathrm{OPT}$$

We first argue that for any sequence $\{y_t\}_{t=1}^T$ such that $\forall t \; a_t + y_t \in \mathcal{A}$ we have (from Proposition 3 in [30]):

$$\sum_{t=1}^{T} S(a_t + y_t) - S(a_t) \geq d_{\min} \sum_{t=1}^{T} S(y_t) \tag{1}$$

For a detailed proof, see the proof of Proposition 3 in [30]. We give here only an intuitive explanation. Let $d_i = \partial_i S(2a_{max} \cdot \mathbf{1})/\partial_i S(\mathbf{0})$, such that $d_{\min} = \inf_i d_i$. A natural implication of submodularity is that S is also an element-wise concave function, as every allocation will produce diminishing returns with the increase of resources allocated. This implies that for every $k \in \mathbb{R}_+$, $i \in [N]$,

$$S(a + ke_i) - S(a) \leq k\partial_i S(a)$$

$$S(\mathbf{0} + ke_i) - S(\mathbf{0}) \leq k\partial_i S(\mathbf{0})$$

From this we note that:

$$\frac{S(a + ke_i) - S(a)}{k} \geq \frac{S(a + me_i) - S(a)}{m} \quad \forall m \in (0, k]$$

$$\frac{S(\mathbf{0} + ke_i) - S(\mathbf{0})}{k} \leq \frac{S(\mathbf{0} + me_i) - S(\mathbf{0})}{m} \quad \forall m \in (0, k]$$

This can be derived from coordinate-wise concavity since the difference quotient of S is non-decreasing, as explained in the proof of Proposition 3 in [30].

Putting these inequalities together and using the DR-submodularity of S with $S(\mathbf{0}) = 0$ we get

$$\frac{S(a + ke_i) - S(a)}{S(ke_i)} \geq \frac{S(a_{\max}) - S(a_{\max} - ke_i)}{S(ke_i)} \geq \lim_{m \to 0} \frac{S(a_{\max}) - S(a_{\max} - me_i)}{S(me_i)} \geq d_i$$

Using the notation $a'_{t,i} = a_t + y^{1:i-1}$ and $a''_{t,i} = a'_{t,i} + y_i e_i$, we prove the inequality for each summand of Equation (1), which then follows

$$\begin{aligned}
S(a_t + y_t) - S(a_t) &= \sum_{i=1}^{N} S(a_t + y^{1:i}) - S(a_t + y_t^{1:i-1}) \\
&= \sum_{i=1}^{N} S(a'_{t,i} + y_i e_i) - S(a'_{t,i}) \\
&\geq \sum_{i=1}^{N} S(a''_{t,i}) - S(a''_{t,i} - y_i e_i) \\
&\geq \sum_{i=1}^{N} S(a_{\max}) - S(a_{\max} - y_i e_i) \\
&\geq \sum_{i=1}^{N} d_i S(y_i e_i) \\
&\geq \sum_{i=1}^{N} d_{\min} S(y_i e_i) \\
&\geq d_{\min} S(y)
\end{aligned}$$

Finally, using Equation (1) with $\{y_t\}_{t=1}^T = \{a_*\}_{t=1}^T$ we get

$$\begin{aligned}
\sum_{t=1}^{T} S(a_* + a_t) - S(a_t) &= \sum_{t=1}^{T} S(a_* + a_t) - S(a_*) + S(a_*) - S(a_t) \\
&\geq d_{\min} \sum_{t=1}^{T} S(a_t) + T \cdot \mathrm{OPT} - \sum_{t=1}^{T} S(a_t) \\
&= T \cdot \mathrm{OPT} - (1 - d_{\min}) \sum_{t=1}^{T} S(a_t) \\
&\geq T \cdot \mathrm{OPT} - (1 - d_{\min}) \left[\sum_{t=1}^{T} S(a_* + a_t) - S(a_t) \right] \\
&\geq (2 - d_{\min})^{-1} \cdot T \cdot \mathrm{OPT}
\end{aligned}$$

8. This step is similar to Theorem 7 in [4]

We first use the following result applied to the queried pairs of points $\{(a_t, a'_t)\}_{t=1}^T$

$$\begin{aligned}
\sum_{t=1}^{T} 2C_t &= \sum_{t=1}^{T} 2(2B + \lambda^{-1/2} \sqrt{\beta(\epsilon, \delta/2, t+1)}) \sigma_{t+1}^{SS'}((a_t, a'_t)) \\
&\leq \sum_{t=1}^{T} 2(2B + \lambda^{-1/2} \sqrt{\beta_t}) \sigma_{t+1}^{SS'}((a_t, a'_t)) \\
&\leq 2(2B + \lambda^{-1/2} \sqrt{\beta_T}) \sum_{t=1}^{T} \sqrt{4(T+2)\gamma_T^{SS'}} \\
&= \mathcal{O}(\sqrt{\beta_T T \gamma_T^{SS'}})
\end{aligned}$$

Where $\sigma_t^{SS'}$ is defined in Appendix A.3 and $a'_t \in \arg\max_{a' \in \mathcal{M}_t} C_t(a_t, a')$ must be queried with a_t, where $\mathcal{M}_t = \{(a_t^1, \dots, a_t^{i-1}, 0, a_t^{i+1}, \dots, a_t^N) \mid i \in [N]\}$. The last inequality comes from the definition of $\sigma_t^{SS'}$, $\gamma_T^{SS'}$ and the bound $k(a, a') \leq 1 \ \forall a, a' \in \mathcal{A}$, knowing that the history $\{(a_\tau, a'_\tau)\}_{\tau=1}^t$ has been queried.

This concludes the proof. □

C.3 Efficient formulation of the optimistic marginal contribution

In this section, we derive an efficient formulation for solving the optimization problem of the optimistic marginal contribution, and this new formulation can be efficiently solved using numerical optimizers.

Definition 6.2 (Optimistic Marginal Contribution)**.** For player i at time t, we define its optimistic marginal contribution as:

$$\overline{\text{marg}}_t^i(a^i, a^{-i}) = \max_{\tilde{S} \in \mathcal{B}_t} \left[\tilde{S}(a^i, a^{-i}) - \tilde{S}(0, a^{-i}) \right]$$

Decomposing the optimization problem of the optimistic marginal contribution we get

$$\begin{aligned} \max_{\tilde{S}} \quad & \tilde{S}(a^i, a^{-i}) - \tilde{S}(0, a^{-i}) \\ \text{such that} \quad & \tilde{S} \in \mathcal{B}_S \\ & \Lambda_t(\tilde{S}) \geq c_{t,\epsilon,\delta} \end{aligned} \tag{2}$$

Now, suppose we are given a solution $\overline{S}$ of Equation (2). We denote the realizations on the history of actions as $Z_{0:t} \doteq \overline{S}(a_{0:t})$ and further note $z \doteq \overline{S}(a)$ the realization for the actions a. We note that in this case, $\Lambda_t(\overline{S})$ is only a function of the realizations. Any function with the same fitting on $\{a_0, ..., a_t, a\}$ will therefore also satisfy the likelihood ratio condition. The condition $\overline{S} \in \mathcal{B}_S$ is a condition on the norm of $\overline{S}$ in the RKHS $\mathcal{H}_k$. Consider the solution S_0 of the following minimum-norm interpolation problem:

$$\begin{aligned} \max_{\tilde{S}} \quad & \langle \overline{S}, \overline{S} \rangle_{\mathcal{H}_k} \\ \text{such that} \quad & \tilde{S}([a_{0:t}^\top \ a]^\top) = \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix} \end{aligned} \tag{3}$$

This problem is quite simple to solve using common tools of optimization theory. To simplify the notations, we first define the following elements:

The feature vector of the actions:

$$\Phi_{0:t,a}(\cdot) = \begin{bmatrix} k(\cdot, a_0) \\ \vdots \\ k(\cdot, a_t) \\ k(\cdot, a) \end{bmatrix}$$

The corresponding kernel matrix $K_{0:t,a}$, where $K_{0:t} = \{k(a_i, a_j)\}_{i,j=0}^t$:

$$K_{0:t,a} = \begin{bmatrix} K_{0:t} & (k(a_\tau, a))_{\tau=0}^t \\ (k(a_\tau, a))_{\tau=0}^t{}^\top & k(a, a) \end{bmatrix}$$

Using the Representer's theorem [31], we can express the result of Equation 3 as a linear combination of the observed features: $S_0 = \alpha^\top \Phi_{0:t,a}$, where $\alpha \in \mathbb{R}^{t+2}$. Moreover, the condition becomes linear and we finally get a least squares solution problem with solution $\hat{\alpha}$ such that:

$$\Phi_{0:t,a}([a_{0:t}^\top\ a]^\top)\hat{\alpha} = \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix} \Leftrightarrow \hat{\alpha} = \Phi_{0:t,a}([a_{0:t}^\top\ a]^\top)^+ \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix}$$
$$\Leftrightarrow \hat{\alpha} = K_{0:t,a}^{-1} \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix}$$

Where in the last step we note that $\Phi_{0:t,a}([a_{0:t}^\top\ a]^\top) = K_{0:t,a}$ by definition, and assume $K_{0:t,a}$ to be invertible[8]. A^+ denotes the pseudoinverse of a matrix A.

As we note that by definition $\|S_0\|_{\mathcal{H}_k} \leq \|\overline{S}\|_{\mathcal{H}_k}$, $\overline{S} \in \mathcal{B}_S$ implies $S_0 \in \mathcal{B}_S$. On the contrary, for any set $[Z_{0:t}^\top\ z]^\top$ such that $S_0 \notin \mathcal{B}_S$, we cannot find any corresponding function $\tilde{S}$ satisfying Equation 2. Now plugging-in the above solution $\hat{\alpha}$ we get:

$$S_0 \in \mathcal{B}_S \quad \Leftrightarrow \quad \langle S_0, S_0 \rangle_{\mathcal{H}_k} = \hat{\alpha}^\top K_{0:t,a} \hat{\alpha} = \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix}^\top K_{0:t,a}^{-1} \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix} \leq B^2$$

This is the motivation for the following efficient formulation of the optimistic marginal optimization problem (as a function of a):

$$\begin{aligned} \max_{Z_{0:t} \in \mathbb{R}^{t+1}, z \in \mathbb{R}} \quad & z - \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix}^\top D \\ \text{such that} \quad & \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix}^\top K_{0:t,a}^{-1} \begin{bmatrix} Z_{0:t} \\ z \end{bmatrix} \leq B^2 \\ & \Lambda_t(Z_{0:t} \mid \mathcal{D}_t) \geq c_{t,\epsilon,\delta} \\ \text{where} \quad & D = K_{0:t,a}^{-1} \Phi_{0:t,a}(0, a^{-i}) \end{aligned} \tag{4}$$

Equivalence with Equation (2) is shown by the following argument:

1. Any solution of Equation (2) corresponds to a solution to Equation 4. This has been shown by the discussion on $\overline{S}$ and S_0 above.

2. Any solution of Equation (4) corresponds to a solution to Equation 2. In that case we can always extract the function $\overline{S} = \hat{\alpha}^\top \Phi_{0:t,a}$, and all the conditions are satisfied, while for the objective we use $z = \overline{S}(a)$ the realization of a and $\overline{S}(0, a^{-i}) = \hat{\alpha}^\top \Phi_{0:t,a}(0, a^{-i})$.

Moreover, this new formulation is convex and can then be solved efficiently using numerical optimizers. Our implementation uses Casadi [32] with the IPOPT optimizer.

[8] We note that k is positive semidefinite, implying that $K_{0:t,a}$ is a positive semidefinite matrix, which in practice with numerical imprecisions can be seen as positive definite with potentially high condition number. In our implementation, we use the regularized $K_{0:t,a} + \epsilon I$ for some $\epsilon > 0$, hence ensuring the matrix is positive definite and thus invertible (with stable numerical inversion)