Automated Design of Affine Maximizer Mechanisms in Dynamic Settings

Michael Curry, Vinzenz Thoma, Darshan Chakrabarti, Stephen McAleer, Christian Kroer, Tuomas Sandholm, Niao He, Sven Seuken

1Harvard University
2University of Zurich
3ETH Zurich
4ETH AI Center
5Columbia University
6Carnegie Mellon University, Computer Science Department
7Optimized Markets, Strategy Robot, Strategic Machine

Abstract

Dynamic mechanism design is a challenging extension to ordinary mechanism design in which the mechanism designer must make a sequence of decisions over time in the face of possibly untruthful reports of participating agents. Optimizing dynamic mechanisms for welfare is relatively well understood. However, there has been less work on optimizing for other goals (e.g. revenue), and without restrictive assumptions on valuations, it is remarkably challenging to characterize good mechanisms. Instead, we turn to automated mechanism design to find mechanisms with good performance in specific problem instances. We extend the class of affine maximizer mechanisms to MDPs where agents may untruthfully report their rewards. This extension results in a challenging bilevel optimization problem in which the upper problem involves choosing optimal mechanism parameters, and the lower problem involves solving the resulting MDP. Our approach can find truthful dynamic mechanisms that achieve strong performance on goals other than welfare, and can be applied to essentially any problem setting—without restrictions on valuations—for which RL can learn optimal policies.

1 Introduction

Dynamic mechanism design studies sequential decision-making problems, where decisions are based on the self-reported preferences of agents. A typical model is that the environment consists of a Markov decision process (MDP), and the mechanism controls the process given reported utilities by the agents. This has important applications, such as ad auctions or more generally online pricing (e.g. Bergemann and Välimäki 2019) but also problems of decentralised decision making in RL (e.g. Chang et al. 2020).

Much work in dynamic mechanism design has focused on maximizing welfare (Athey and Segal 2013; Nisan et al. 2007a; Lyu et al. 2022b) subject to strategyproofness (there should be no incentive for untruthful reports by agents). Some other work considers different goals, notably revenue (Bergemann and Välimäki 2010; Kakade, Lobel, and Nazerzadeh 2013; Hajiaghayi, Kleinberg, and Parkes 2004; Hajiaghayi, Kleinberg, and Sandholm 2007) but needs to make restrictive assumptions about the space of agent types. Work on dynamic mechanism design, for general goals and for broad spaces of agent types, is much more limited.

Dynamic mechanism design includes as a special case static mechanism design, and here the situation is similar. To maximize welfare while ensuring strategyproofness, one can use the celebrated and well-understood Vickrey-Clarke-Groves (VCG) mechanism (Vickrey 1961; Clarke 1971; Groves 1973). For optimizing revenue, Myerson (1981) completely settles the question under the restrictive assumption that agents’ types are single-dimensional (essentially, that there is only one type of item up for sale); beyond this there has been little progress except for very specific problem instances (Yao 2017). But for static settings, automated mechanism design (AMD) (Conitzer and Sandholm 2002; Sandholm 2003; Curry, Sandholm, and Dickerson 2023) has been used: this is a data-driven search through some class of mechanisms in order to find one that performs well while satisfying the constraints of strategyproofness and individual rationality. Automated mechanism design has in some cases found the highest-performing mechanisms known so far, and can recover optimal mechanisms in special cases where they are known (Duetting et al. 2019; Ivanov et al. 2022; Shen, Tang, and Zuo 2019).

Given the successes of automated mechanism design for static problems, it is surprising that its use for dynamic problems is relatively underexplored.

Our Contributions

Our present paper develops automated dynamic mechanism design techniques which can be applied to a very broad range of problems. In particular, we consider mechanism design on general MDPs. Our model works with many loss functions (including revenue but also other domain-specific loss functions), not just the easier goal of welfare, and we do not assume one-dimensional agent types. Our assumptions are sufficiently general to capture essentially all of static
multi-parameter mechanism design as a special case\textsuperscript{1}.

Optimal mechanism design over all possible mechanisms entails the very difficult problem of computing equilibria in imperfect information games, to understand whether or not any agent has any incentive to deviate from truthful reporting. Inspired by prior work for static mechanism design, we sidestep this issue by focusing on the class of affine maximizers (AMAs), which we define on MDPs.\textsuperscript{2} These mechanisms are always strategyproof.

We justify this restriction in two ways. First, as mentioned, our problem assumptions capture multi-parameter mechanism design—but finding optimal mechanisms in this setting has proven extremely difficult, so it makes sense to search within a more tractable class of mechanisms. Second, our framing of the problem is broad enough that it captures problem instances where Roberts’ theorem (Roberts 1979) applies, which states that under general agent type spaces, only affine maximizers can be strategyproof.

We frame the search for a high-performing dynamic affine maximizer mechanism as a bilevel optimization problem, where the outer problem consists of choosing weights and (possibly state-dependent) boosts—the AMA parameters—to minimise a given loss function. The inner problem consists of learning to control an MDP to maximise affinely-transformed social welfare (the definition of an affine maximizer), given the weights, boosts, and agents’ type reports (see equation 1 below). The derivatives of this inner problem do not exist at many points: however, we show that for the important case of revenue, the expected loss over the distribution (with a continuous density) of agent valuations is differentiable.

We solve the inner problem via (possibly regularized) linear programming. We also propose a variety of ways to solve the outer problem: by grid search, by differentiating through the regularized LP, or by using zeroth-order methods to approximate the LP gradient. These latter two approaches explicitly or implicitly smooth the objective and avoid the problem of nonexistent derivatives. In experiments on several dynamic mechanism design settings, such as sequential auctions, task scheduling and navigating a gridworld, our approaches result in truthful mechanisms that outperform the VCG baseline.

2 Related Work

Maximizing Welfare in Dynamic Mechanisms

Athey and Segal (2013) consider a dynamic mechanism design setting where agents update their beliefs over time, and where the goal is an efficient and budget-balanced outcome. Parkes (Nisan et al. 2007b) describes a dynamic mechanism design setting where the focus is on agents who may arrive and depart at different periods. Both of these approaches simply assume an optimal policy is available. More recently, Lyu et al. (2022b) presents a model for learning this policy in an offline RL setting; another work focuses on the online case (Lyu et al. 2022a). Chang et al. (2020) consider a dynamic VCG mechanism for decentralised RL where agents bid on the MDP transitions. Bergemann and Välimäki (2010) presents a VCG-like “dynamic pivot mechanism”. There are different modeling choices and goals in each of these approaches, but the common theme is that the allocation problem involves making decisions on some MDP after observing agent reports. All of these papers consider a dynamic analogue to the VCG mechanism, i.e. the mechanism designer acts according to the welfare-optimal policy and charges agents their externalities to ensure incentive compatibility. This is in contrast to our work, where we are concerned with goals beyond welfare.

Dynamic Mechanism Design for Goals Other Than Welfare

There is some existing work in this direction as well, with a particular focus on revenue. Puvan, Segal, and Toikka (2014) and Kukade, Lobel, and Nazerzadeh (2013) both consider cases where the private information about the value of the item is 1-dimensional. This allows for a Myerson-style analysis of the actual profit-maximizing mechanism. Other work considers optimal selling of items to agents who arrive and depart over time from the perspective of optimal stopping, but again considers single-parameter item valuations (Hajiaghayi, Kleinberg, and Parkes 2004; Hajiaghayi, Kleinberg, and Sandholm 2007; Kleinberg 2005). Bergemann and Välimäki (2019) surveys other results that make similar assumptions for tractability. Pai and Vohra (2008) considers a similar setting and finds the optimal Bayes-Nash incentive compatible mechanism. Still other work considers settings where the mechanism designer may update the mechanism online over time (e.g. by changing reserve prices) (Shen et al. 2017) and where bidders may even strategically attempt to manipulate the learning process (Amin, Rostamizadeh, and Syed 2014). Overall, these approaches are restrictive in terms of their assumed value model and frequently focus on analytic results, while our computational approach allows more general values and loss functions.

Preference Elicitation from Multiple Agents and Multistage Mechanisms

Another line of work considers iterative preference elicitation (Conen and Sandholm 2001; Sandholm and Boutilier 2006): based on past agent reports, the mechanism can query what preference information it needs most. Existing approaches make use of an analogy to query learning (Zinkevich, Blum, and Sandholm 2003; Blum et al. 2004; Lahαιe and Parkes 2004), or leverage machine learning (Soumallas et al. 2023; Weissteiner et al. 2023; Brero, Lubin, and Seuken 2019). In this context, Sandholm, Conitzer, and Boutilier (2007) use automated mechanism design to first find good mechanisms (for general type spaces) and then convert them into multistage mechanisms. While these app-
proaches are typically focused on making a single final decision, but eliciting agent’s preferences over multiple stages, our automated dynamic mechanism design approach is concerned with making a sequence of decisions over time, while eliciting preferences once.

**Static Automated Mechanism Design**

Due to the difficulty of analytically finding optimal mechanisms, a number of works have instead attempted to treat static mechanism design as a computational optimization problem, starting with Conitzer and Sandholm (2002) and Sandholm (2003), and learn good mechanisms from samples, starting with Likhodedov and Sandholm (2004). One line of work makes use of static affine maximizers and achieves good performance in multi-item multi-bidder auctions (Sandholm and Likhodedov 2015; Curry, Sandholm, and Dickerson 2023; Duan et al. 2023). There is also a line of learning theory research on choosing the AMA parameters given samples from the valuation distribution (Balcan, Sandholm, and Vitercik 2016, 2018; Balcan et al. 2021). Another direction is to start with a potentially non-strategyproof mechanism and iteratively modify it to improve strategyproofness. This is known as *incremental mechanism design* (Conitzer and Sandholm 2007). One line of work in this direction makes use of rich function approximators to learn mechanisms. Duetting et al. (2019) presents one influential direction, which uses neural networks to optimize revenue and a penalized loss to approximately enforce strategyproofness, with many followups (Curry et al. 2021, 2020; Ivanov et al. 2022; Rahme, Jelassi, and Weinberg 2021; Rahme et al. 2021). Shen, Tang, and Zuo (2019) presents an alternative approach for single-bidder settings which can cope with broader classes of utility functions. As mentioned above, the success of these techniques in static settings is our motivation to develop such approaches for the dynamic setting.

**3 Preliminaries**

Below, we describe our mechanism design problem. The nature of the problem and our mechanism design desiderata motivate our choice to restrict attention to affine maximizer mechanisms. We then describe in more detail how these mechanisms work in a dynamic setting.

**Formal Model of Problem Setting**

**Environment and policy/allocation rule** Consider some MDP $\mathcal{M} = (S, A, P)$ (with reward not yet specified), where $S$ is a set of states, $A$ is a set of actions, and $P$ is a transition function. There are $n$ agents, each with their own reward function $r_t : S \times A \rightarrow \mathbb{R}$ drawn from a distribution with density $f_t$. We emphasize that these agents are not themselves taking actions in the MDP—this is done by the mechanism. Their only choice will be which rewards to report to the mechanism. We assume the mechanism designer wants to minimize some loss function (which will often be the negative of some objective to be maximized) $L$ in expectation over the $f_t$, which in general depends on the agents’ rewards and the chosen policy $\pi$ on the MDP. The latter corresponds to the allocation rule in traditional mechanism design.

**Mechanism Design Desiderata**

**Incentive compatibility and payments** In order to achieve its goal, the mechanism will need access to the true $r_t(s, a)$. However, in general, we should expect the agents to misreport their reward function if they think it will benefit them. Thus, we will allow for some side payments to be made based on the MDP’s solution and the agents’ reports, in order to ensure that there is no incentive to misreport, that is, making the mechanism incentive compatible (IC) or strategyproof. (Such payments only exist for some choices of $\pi$.) We assume agent utility is quasilinear, that is, positive payments just correspond to negative reward.

**Individual rationality** We also want to guarantee individual rationality (IR), meaning that agents should not be charged so much that they receive negative utility and would be better off not participating in the mechanism.

**Remark.** The mechanism designer’s goal may also relate to the payments. For example, a canonical mechanism design goal is to maximize revenue—in our setting, choose a policy $\pi^*$ such that payments can be made as high as possible while still ensuring IC and IR.

**Background on Affine Maximizers**

As motivated above our goal is to choose some $\pi$ such that IC/IR side payments can be constructed, while also performing as well as possible on the mechanism designer’s higher-level objective.

Unlike prior work (e.g., Kakade, Lobel, and Nazerzadeh 2013; Bergemann and Välimäki 2010), we do not make any assumptions about the structure of the reward functions. Our setting is therefore general enough to incorporate many hard problems such as optimal multi-item mechanism design, and is thus at least as hard as those. Therefore hoping to get the truly best-performing $\pi$, even only in an infinite-sample/asymptotic sense, is too much. Thus it is appropriate to restrict attention to a more tractable class of mechanisms.

Also, our problem setting is general enough to include situations where a result known as Roberts’ theorem applies (Roberts 1979). It states that under certain conditions (arbitrary rewards, at least 3 outcomes), the only allocation rules that can possibly have IC payments take the form of affine maximizers.

Below, we give the standard definition of affine maximizer mechanisms in terms of the allocation and payment rules, modified for our dynamic problem setting.

**Definition 3.1** (Affine maximizers). Given so-called weights $w \in \mathbb{R}_n^+$ and boosts $b \in \mathbb{R}^{|S| x |A|}$, a dynamic affine maximiser mechanism (AMA) takes reported reward functions $r \in \mathbb{R}^{|S| x |A|}$ and returns a policy $\pi^*(w, b, r)$ on the MDP that maximizes the affine social welfare $\text{asw}^*(\pi)(w, b, r)$ where

\[
\text{asw}^*(\pi)(w, b, r) = \mathbb{E}_{\pi} \left[ \sum_{t=0}^{T} \left( \sum_{i=1}^{n} w_i r_t(s_i, a_i) \right) + b(s_t, a_t) \right]
\]

Defining $\text{asw}(w, b, r) = \text{asw}^*(\pi^*(w, b, r))(w, b, r)$ as the maximum affine social welfare for reports $r$, the resulting payment is then
\[ p_i(w, b, r) = \frac{1}{w_i} \left( \text{asw}^{(-1)}(w, b, r) \right) \]

\[ = \left( E_{\pi^*(w, b, r)} \left[ \sum_{t=0}^{T} \left( \sum_{j \neq i} w_j r_j(s_t, a_t) \right) + b(s_t, a_t) \right] \right) \]

where \( \text{asw}^{(-1)}(w, b, r) \) defined as

\[ \max_{\pi} E_{r \sim f} \left[ \sum_{t=0}^{T} \left( \sum_{j \neq i} w_j r_j(s_t, a_t) \right) + b(s_t, a_t) \right] \]

is the maximum affine social welfare, when disregarding \( i \).

No matter the choice of weights and boosts, every resulting AMA is strategyproof ex-post—after learning the rewards of the other agents, reporting truthfully is a dominant strategy—and IR in expectation over the MDP trajectories. The proof can be found in Appendix A.

4 Dynamic Mechanism Design as Bilevel Optimisation

The problem of searching for a performant mechanism within the class of AMAs can be naturally formulated as a stochastic bilevel optimization as follows\(^3\):

\[ \min_{w, b} E_r \left[ \mathcal{L}(\pi^*, w, b) \right] \text{ s.t. } \pi^* \in \arg \max_{\pi} E_{s_t, a_t \sim \pi} \left[ \sum_{t=0}^{T} \left( \sum_{i=1}^{n} w_i r_i(s_t, a_t) \right) + b(s_t, a_t) \right] \forall r \]

Given the bilevel structure, we can think of the problem as a game between a leader and a follower.

- The leader knows only the joint distribution \( f = \prod_i f_i \) from which the set of reward functions are drawn, and chooses weights \( w_i \) and boosts \( b(s, a) \).

- For any draw of \( r_i \) and the weights and boosts, the follower acts optimally ("best responds") in the MDP according to the AMA objective.

To clarify (and to contrast with many models of mechanism design where the agents making reports are treated as followers): the leader and follower are only "notional". In reality, there is only one mechanism designer. We nevertheless speak in terms of a "leader" and "follower" because in order for strategyproofness to be attained, some component of the system must successfully maximize affine social welfare, which is a goal distinct from the true goal of the mechanism designer.

In general, the problem above—bilevel optimization, with stochasticity in the leader’s objective—is quite difficult. Indeed, the case of revenue-maximisation in one-round auctions with multiple goods, which is a very special case of our much more general problem, remains essentially unsolved beyond a few very simple special cases (Yao 2017). Therefore finding a globally optimal solution to the above problem is too much to hope for, but we show that derivatives exist for important expected loss functions, which enables us to use gradient-based optimization techniques to find local optima.

We then consider three complementary methods for optimizing the AMA parameters: random grid search, zeroth-order methods to approximate the derivatives, and differentiation through a smoothed LP.

Existence of derivatives

So far, we have not made any assumptions about the loss function. A very natural desideratum would be that \( \mathcal{L}(\pi^*(w, b, r), w, b) \) is differentiable, so that we can perform stochastic gradient descent. However, in general this is not true. This is due to the fact that \( \pi^*(w, b, r) \) is in general not even continuous in \( w, b \)—since the optimal policy on an MDP is always going to be deterministic—and therefore neither is \( \mathcal{L} \). However, if \( \mathcal{L} \) has a certain shape, which is the case for the loss functions we consider in this work, we can prove that the relaxed condition that \( E_r[\mathcal{L}(\pi^*(w, b, r), w, b)] \) is differentiable.

**Theorem 4.1.** Let \( \mathcal{L} \) be a loss function for the problem in Equation (1) and assume it can be decomposed as follows

\[ \mathcal{L}(\pi^*(w, b, r), w, b) = \text{asw}(w, b, r) + \sum_{k=1}^{K} a_k g_k(\pi^*(w, b, r), r) \]

where it holds for all \( k \) that \( g_k = \mathcal{O}(\|r\|_\infty) \). Assume further that the support of \( r \) is compact or \( f \) decays sufficiently quickly such that \( E[\|r\|_\infty] \) exists. Then \( E_r[\mathcal{L}(\pi^*(w, b, r), w, b)] \) is differentiable almost everywhere.

**Proof Idea.** The full proof is in Appendix B. It consists of two parts. First we show that asw is Lipschitz continuous in \( w, b \). This essentially follows because it is the maximum over a set of functions which are linear in \( w, b \). For \( g_k \) we argue that for any \( w, b \), the set of rewards for which two alternative policies give the same affine social welfare, lie on a hyperplane. Changing the weights and boosts may change the optimal policy, and therefore may result in differing social welfare for some subset of the rewards. We would like to bound the probability mass of the set of rewards where this can happen. Indeed, the rewards where the optimal policy may change, all lie in between the aforementioned hyperplanes. The distance between the hyperplanes depends on the change in \( w \) and \( b \), so the mass can be bounded—assuming the probability density decays sufficiently fast—which results in a local Lipschitz constant. Differentiability follows from Rademacher’s theorem and holds equivalently for the sum of these terms because of the linearity of the expectation and derivative.

In Section 5 we will study two loss functions in particular: revenue and makespan. The revenue of a dynamic AMA
mechanism is given by
\[
\text{rev}(w, b, r) = \sum_{i=1}^{n} p_i(w, b, r) = -\sum_{i=1}^{n} \left( \frac{1}{w_i} \text{asw}(w, b, r) \right) + \text{sw}(\pi^*(w, b, r))(r)
+ \sum_{i=1}^{n} \frac{1}{w_i} \text{asw}(-1)(w, b, r)
\]

Here \( \text{sw} \), the non-transformed social welfare, is just \( \text{asw} \) when weights are all 1 and boosts are all 0, i.e. \( \text{sw}(\pi) = E_{\pi} \sum_{i=0}^{T} \sum_{i=1}^{n} r_i(s_i, a_t) \).

It is easy to verify that revenue fulfills all assumptions of Theorem 4.1.

**Lemma 4.2.** The expected revenue is differentiable almost everywhere.

**Proof.** Revenue is a essentially a sum of the three terms \( \text{asw} \), \( \text{sw} \) and \( \text{asw}(-1) \). The first \( \text{asw} \) is directly as stated in the assumptions of Theorem 4.1. For \( \text{asw}(-1) \) the proof extends canonically as it equivalently is the maximal affine social welfare, albeit for a smaller set of agents. Last but not least \( \text{sw}(\pi^*(w, b, r))(r) \) is a function of only \( \pi^*(w, b, r) \) and \( r \), for which it holds that
\[
\text{sw}(\pi^*(w, b, r))(r) = E_{\pi} \left[ \sum_{i=0}^{T} \sum_{i=1}^{n} w_i r_i(s_i, a_t) + b(s_i, a_t) \right]
\leq T n\|r\|_{\infty}
\]

thereby concluding the proof.

Once we have introduced the task scheduling problem we will analogously show that makespan also satisfies the assumptions of Theorem 4.1.

**Linear Programming Formulation**

We now describe the linear-programming formulation for an MDP, which can be either infinite horizon or episodic with states partially ordered by time (that is, the time step is encoded in the state) (Altman 1999). In particular, we suppose the follower solves the following LP to find the optimal state-action occupancy measure \( \nu_{\pi^*(w, b, r)} \), which determines the revenue (or whichever loss function is chosen):
\[
\begin{aligned}
& \max \sum_{s \in S, a \in A} \left( \sum_{i} w_i r_i(s, a) + b(s, a) \right) \nu(s, a) \text{ s.t.} \\
& \sum_{a \in A} \nu(s, a) = \sum_{s' \in S, a' \in A} P(s'|s, a') \nu(s', a') + \mu_0(s) \quad \forall s \in S \\
& \nu(s, a) \geq 0 \quad \forall s \in S, a \in A
\end{aligned}
\]

where \( \mu_0 \) denotes the initial state distribution.

Note that if \( \mathcal{L} \) were itself differentiable, we could take the gradient inside the expectation and apply the implicit function theorem to get \( \nabla_{w, b} \mathbb{E}_{r} [\mathcal{L}(\pi^*, w, b)] = \mathbb{E}_{r} [\nabla_{w, b} \mathcal{L}(\pi^*, w, b) + \nabla_{w, b} \mathcal{L}(\pi^*, w, b) \nabla_{\pi^*} \mathcal{L}(\pi^*, w, b)] \).

Then we could estimate the gradient of the expected value from a sum of gradients for different sampled \( r \). Instead as we show in Appendix C, for revenue we can even compute the partial derivatives of \( \mathcal{L} \) with respect to \( w, b \) analytically. However, because \( \mathcal{L} \) is not differentiable (since \( \pi^* \) is not), the Dominated Convergence Theorem does not hold and we cannot in fact exchange gradient and expected value. Instead we propose two alternatives to compute the gradient of the expected value. First using zeroth-order estimates and second introducing regularization, which makes the optimal policy in the inner problem unique \( \mathcal{L} \) differentiable and thus actually allows us to estimate the gradient from samples. The pseudocode of our approach is given by Algorithm 1.

**Zeroth-order methods** For bilevel problems, Sow, Ji, and Liang (2022) present a zeroth-order approach, which we adapt to our own setting. The key observation of their approach is that the derivative of the leader’s objective is the sum of two partial derivatives. One of these, the derivative of the leader’s objective with respect to their own solution, is usually easy to evaluate. The other requires differentiating through the follower’s best-response map and inverting a Jacobian, which is challenging—and this second portion can be separately estimated using zeroth-order perturbations.

Zeroth-order methods, due the use of random perturbations, also implicitly smooth the function (Duchi, Bartlett, and Wainwright 2012), so that when using them there is no further need for regularization to ensure that derivatives can be estimated.  

**Regularized linear program** As an alternative method, we propose regularizing our problem to get a smooth surrogate of the derivative. We add a small entropy regularizer to the follower’s objective.  

The result is now a convex program (see 13 in the appendix): When the objective is strongly convex, the solution map \( \nu_{\pi^*} : (w, b) \mapsto \nu_{\pi^*(w, b, r)} \) is smooth. Therefore, derivatives of revenue can now be estimated from the gradients at sampled type profiles, calculated using reverse-mode automatic differentiation (Agrawal et al. 2020, 2019).

Adding a small amount of regularisation does not change the follower’s problem significantly. As has been shown by Weed (2018), the distance between the solutions to the regularized and unregularized inner problem decays exponentially fast in the regularisation constant \( \alpha \), for sufficiently small \( \alpha \).

Under certain assumptions we claim this convergence translates to the outer problem, so that choosing a small \( \alpha \) ensures the objective is not disturbed too much.

\footnote{They have been proposed in certain related (static) settings (Bichler et al. 2021; Martin and Sandholm 2023) to cope with similar problems related to nonexistence of derivatives.}

\footnote{A technique equivalent to such regularization has also been used to deal with a similar issue in first-order computation of equilibria of non-truthful static auctions (Kohring, Pieroth, and Bichler 2023).}
Algorithm 1: Gradient-based Dynamic Mechanism Design

**Input:** MDP $\mathcal{M}$, number of agents $n$, loss $\mathcal{L}$, number of initialisations $m$

1: Initialize: $w \in \mathbb{R}^n$, $b \in \mathbb{R}^{|S| \times |A|}$
2: $(w_i, b_i)_{1 \leq i \leq m} \leftarrow \text{grid search} (\mathcal{M}, \mathcal{L})$
   \{Multiple starting points to avoid local minima\}
3: for num_iterations do
   4: for $i = 1$ to $m$ do
   5:   $(w_i, b_i) \leftarrow (w_i, b_i) + \gamma \text{gradient step} (\mathcal{M}, w_i, b_i)$
   \{Zeroth or first order gradient estimate\}
   6: end for
   7: end for
8: return $\arg\min_{i \in \{1, \ldots, m\}} \mathbb{E}[\mathcal{L}(w_i, b_i)]$

**Theorem 4.3** (Pointwise convergence of regularised loss). Assume that $\mathcal{L}(w, b, r)$ can be represented as an inner product between a vector $q \in \mathbb{R}^{|S| \times |A|}$ and the state-action occupancy measure $\nu$, such that $\|q\|_\infty \leq C(\|r\|_\infty)$. Let $L_{\alpha}(w, b, r)$ denote the loss achieved with the optimal policy $\pi^*(w, b, r)$ for the regularised problem (see equation 13 in Appendix E). Assume further that $\mathbb{E}[\|r\|_\infty] < \infty$, then

$$\lim_{\alpha \to 0} \mathbb{E}[L_{\alpha}(w, b, r)] = \mathbb{E}[L(w, b, r)]$$

The proof of the Theorem can be found in Appendix D. Given that both affine social welfare and social welfare can be represented as an inner product of $\nu$, Theorem 4.3 implies the convergence of regularised revenue. Similarly, we will later show the same holds for makespan.

**Corollary 4.4.** Let $rev_{\alpha}(w, b)$ denote the revenue achieved with the optimal policy $\pi^*(w, b, r)$. Assume that $\mathbb{E}[\|r\|_\infty]$ exists, then

$$\lim_{\alpha \to 0} rev_{\alpha}(w, b) = rev(w, b)$$

### 5 Experiments

**Methods**

We optimize AMAs in dynamic mechanism design settings on tabular MDPs. We compare three different mechanism design settings with different reward distributions. Our optimization methods consist of a naïve grid search baseline, and two gradient-based methods (using either zeroth-order or regularized gradient estimates).

**Implementation Details**

**Grid search** As a methodological benchmark, we implement a naïve grid search algorithm. We sample 10000 different weights and boosts from a Sobol sequence (Sobol 1967). To assess the expected performance of each draw, we solve the associated dual linear program (LP), as detailed in Equation (2), for 2000 randomly sampled reward profiles.

**Zeroth-order methods** We estimate derivatives using 20 perturbations, sampled from a Gaussian distribution with standard deviation 0.05 per estimate on 20 sampled type profiles. We use a learning rate of 0.1.

**Table 1:** Results for optimizing auction revenue in a sequential sales setting ($n$ agents, $m$ sales) with symmetric uniformly-distributed types. Standard errors were < 0.007. Runtime was < 31 hours for grid search, < 0.2 hours for zeroth-order and < 1.1 hours for first order.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>VCG</th>
<th>Grid</th>
<th>0-order</th>
<th>reg. LP</th>
<th>Best imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>0.0000</td>
<td>0.4902</td>
<td>0.4939</td>
<td>0.4893</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.4999</td>
<td>0.6079</td>
<td>0.6777</td>
<td>0.6755</td>
<td>35.56%</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.8000</td>
<td>0.7977</td>
<td>0.8783</td>
<td>0.8328</td>
<td>9.79%</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.9996</td>
<td>1.0050</td>
<td>1.0078</td>
<td>0.9967</td>
<td>0.83%</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.0000</td>
<td>0.4715</td>
<td>0.3825</td>
<td>0.4168</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.0000</td>
<td>0.6107</td>
<td>0.6446</td>
<td>0.7240</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0.6007</td>
<td>0.7323</td>
<td>0.8875</td>
<td>0.9104</td>
<td>51.54%</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0.9988</td>
<td>0.7142</td>
<td>1.0631</td>
<td>1.0743</td>
<td>7.56%</td>
</tr>
</tbody>
</table>

**Regularized LP.** We compute derivatives with respect to social welfare using the regularized LP, with smoothing parameter $10^{-2}$ except where mentioned. We solve the regularized program using MOSEK (ApS 2023) and use the DiffOpt package within JuMP (Lubin et al. 2023) to differentiate. For the affine social welfare terms in the expected revenue we use the partial derivatives given in Appendix C. For each stochastic gradient step, we sample 20 type profiles and optimize with learning rate $10^{-2}$.

In all cases, when evaluating the objective, we sample 10000 type profiles and do not use regularization. Thus at evaluation time, the LP solution is exactly correct, ensuring strategyproofness. Computational details and hyperparameters are described in appendix F.

**Environments and Results**

**Sequential Sales** We begin with a simple setting in which identical items are sold sequentially to unit-demand bidders. The states consist of a record of who has received the item; the allowed actions are to sell the item to some bidder, or to no one. The welfare-maximizing mechanism thus involves giving the items to the highest bidder, but by altering the boosts, revenue can be increased by sometimes withholding the item. We consider a distribution of type profiles drawn uniformly from $[0, 1]$, with results in Table 1.

We observe that optimizing the boosts can consistently improve performance compared to VCG, especially when there are no tight supply constraints. Intuitively, if there is a large supply of goods, VCG revenue should be low, as agents do not cause much externality on other agents. By setting boosts to effectively withhold goods (like a reserve price), revenue can be increased. We also consider a distribution where agent $i$’s type is uniformly distributed on $[0, i]$. In this setting, we also allow the bidder weights to vary, with results in Table 2. Again, we observe improved performance by optimizing the AMA parameters.

In both settings, the gradient-based approaches generally outperform the random grid search both in terms of results and runtime, in particular for the larger settings, which makes sense considering the high number of dimensions.
Dynamic truthful task scheduling The next setting we consider is a dynamic version of the classic truthful task scheduling problem (Nisan and Ronen 2001). In the static problem, workers report the time it takes them to complete certain tasks. A mechanism then has to assign the tasks and give payments to incentivize truthful reports with the goal of minimizing the makespan of all jobs. We formulate a dynamic version of the truthful task scheduling problem. There are $n$ workers and $T$ tasks. Each round, one of the tasks arrives and has to be assigned. Each worker has a cost vector $\theta_i = (t_{i,1}, \ldots, t_{i,T})$ distributed according to some prior $f_i$, which consists of the times they take to finish each task. Each round $\tau$ the mechanism designer takes an allocative action $x_{\tau} \in \{0, 1\}^n$, s.t. $\sum_{i=1}^n x_{\tau,i} = 1$. At time $\tau$, this causes agent $i$ to receive reward $r_{i,\tau} = -x_{\tau,i}t_{i,\tau}$.

The objective of the leader is to minimize total makespan. For this define $\hat{t}_i$ as the time $t_i$ has already worked on its jobs, when the last task has been assigned in round $T$. The leader’s loss is then given by $\max_{i \in \{1, \ldots , n\}} \left( \left( \sum_{\tau=1}^{T} x_{\tau,i}t_{i,\tau} \right) - \hat{t}_i \right)$.

In order to justify using our approach for minimizing makespan, we need to show that Theorems 4.1 and 4.3 apply. To see that makespan fulfills the assumptions of Theorem 4.1, notice that it is a function of only $t$ (which is $-r$ and $\pi^*(w, b, r)$ and that the total makespan is always bound by $T\|r\|_{\infty}$ Therefore we can conclude that it is differentiable almost everywhere. Further notice that the makespan of a single agent (i.e. without the max operator in front) fulfills the assumptions of Theorem 4.3. Let the maximum difference across all agent-specific makespans between the unregularised and regularised optimal policy be $\epsilon$, then the difference in the total makespan can be at most $2\epsilon$. But $\epsilon \rightarrow 0$ according to Theorem 4.3 which shows the makespan under the regularized optimal policies will converge to the correct makespan as $\alpha \rightarrow 0$.

As a benchmark, we consider a dynamic VCG mechanism that chooses the solution, which minimizes the total work done by the agents—an objective which is not the same as minimizing makespan. (It can be shown, however, that VCG is an $n$-approximation of the optimal static mechanism (Nisan and Ronen 2001)).

For a participant-symmetric valuation distribution (uniform on $[0, 3]$), results are in Table 3. Across all environments, we see an improvement in makespan for the optimized AMAs over the VCG mechanism. We also consider an asymmetric distribution with disutilities distributed on $[0, 3]$ for bidder $i$, with results in Table 4. Across all environments, we see a significant improvement in makespan for the best AMAs. In both settings, gradient-based approaches outperform the naïve grid search both in terms of results and runtime, as was observed in the sequential auction environments.

Navigating a grid with multiple tasks One of the most canonical environments in RL is the gridworld, where

---

Table 2: Results for optimizing auction revenue in a sequential sales setting ($n$ agents, $m$ sales) with asymmetric uniformly-distributed types. Standard errors were < 0.004. Runtime was < 32 hours for grid search, < 0.2 hours for zeroth-order and < 1.1 hours for first order.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>VCG</th>
<th>Grid</th>
<th>0-order</th>
<th>reg. LP</th>
<th>Best imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>0.0000</td>
<td>0.3327</td>
<td>0.3302</td>
<td>0.3665</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.2348</td>
<td>0.3217</td>
<td>0.4123</td>
<td>0.4116</td>
<td>75.55%</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.3089</td>
<td>0.3328</td>
<td>0.4021</td>
<td>0.4508</td>
<td>45.95%</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.3350</td>
<td>0.3355</td>
<td>0.4348</td>
<td>0.4645</td>
<td>38.65%</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.0000</td>
<td>0.3208</td>
<td>0.2544</td>
<td>0.3202</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.0000</td>
<td>0.3304</td>
<td>0.3556</td>
<td>0.4354</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0.2276</td>
<td>0.3431</td>
<td>0.4263</td>
<td>0.4751</td>
<td>108.77%</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0.3193</td>
<td>0.2713</td>
<td>0.3770</td>
<td>0.4976</td>
<td>55.85%</td>
</tr>
</tbody>
</table>

Table 3: Results for minimizing makespan in the dynamic truthful task scheduling setting ($n$ agents, $m$ sales) with symmetric uniformly-distributed types. Standard errors were < 0.02. Runtime was < 16 hours for grid search, < 0.1 hours for zeroth-order and < 4.1 hours for first order. AMA outperforms VCG because the makespan is smaller.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>VCG</th>
<th>Grid</th>
<th>0-order</th>
<th>reg. LP</th>
<th>Best imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>1.0336</td>
<td>1.0326</td>
<td>0.9132</td>
<td>0.9288</td>
<td>-11.65%</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1.0116</td>
<td>1.0142</td>
<td>0.8820</td>
<td>0.9286</td>
<td>-12.81%</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.6111</td>
<td>0.6196</td>
<td>0.5967</td>
<td>0.6184</td>
<td>-2.36%</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0.5662</td>
<td>0.5632</td>
<td>0.5429</td>
<td>0.5538</td>
<td>-4.12%</td>
</tr>
</tbody>
</table>

Table 4: Results for minimizing makespan in the dynamic truthful task scheduling setting ($n$ agents, $m$ sales) with asymmetric uniformly-distributed types. Standard errors were < 0.03. Runtime was < 17 hours for grid search, < 0.1 hours for zeroth-order and < 4 hours for first order. AMA outperforms VCG because the makespan is smaller.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>VCG</th>
<th>Grid</th>
<th>0-order</th>
<th>reg. LP</th>
<th>Best imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>1.8312</td>
<td>1.8260</td>
<td>1.5978</td>
<td>1.6121</td>
<td>-12.75%</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1.9651</td>
<td>1.9837</td>
<td>1.6347</td>
<td>1.6886</td>
<td>-16.81%</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1.4299</td>
<td>1.4426</td>
<td>1.3242</td>
<td>1.3243</td>
<td>-7.39%</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1.4644</td>
<td>1.4729</td>
<td>1.3256</td>
<td>1.3629</td>
<td>-9.48%</td>
</tr>
</tbody>
</table>

---

\footnote{This in contrast to the auction environment where the mechanism could charge payments, because here agents suffer costs for which they need to be compensated.}
an agent deterministically navigates a two-dimensional grid with rewards for reaching certain states (Sutton 2018). We consider the following variant: the mechanism reaches a state for zeroth-order and < 1 hours for first order.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>VCG</th>
<th>Grid</th>
<th>0-order</th>
<th>reg. LP</th>
<th>Best imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>0.7547</td>
<td>1.0607</td>
<td>1.3486</td>
<td><strong>1.5464</strong></td>
<td>104.90%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1.2812</td>
<td>1.4348</td>
<td>1.5575</td>
<td><strong>1.9251</strong></td>
<td>50.25%</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1.8683</td>
<td>1.8729</td>
<td>1.8853</td>
<td><strong>2.3009</strong></td>
<td>23.15%</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2.3563</td>
<td>2.3689</td>
<td>1.9951</td>
<td><strong>2.5989</strong></td>
<td>10.30%</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1.0402</td>
<td>1.0446</td>
<td>1.4935</td>
<td><strong>1.6134</strong></td>
<td>55.11%</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1.4898</td>
<td>1.4904</td>
<td>1.6821</td>
<td><strong>2.0171</strong></td>
<td>35.40%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1.8610</td>
<td>1.8757</td>
<td>1.9427</td>
<td><strong>2.3413</strong></td>
<td>25.81%</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2.2472</td>
<td>2.2133</td>
<td>2.0256</td>
<td><strong>2.5911</strong></td>
<td>15.30%</td>
</tr>
</tbody>
</table>

In Table 5, we present the results for maximising revenue in the gridworld environment (n agents, m × m grid). Standard errors were < 0.05. Runtime was < 30 hours for grid search, < 0.1 hours for zeroth-order and < 0.1 hours for first order.


Curry, M.; Sandholm, T.; and Dickerson, J. 2023. Differentiable Economics for Randomized Affine Maximizer Auctions. In *International Joint Conference on Artificial Intelligence (IJCAI).*


Hajighayi, M.; Kleinberg, R.; and Sandholm, T. 2007. Automated online mechanism design and prophet inequalities. In *AAAI Conference on Artificial Intelligence (AAAI).*


Kleinberg, R. 2005. A Multiple-Choice Secretary Algorithm with Applications to Online Auctions. In *Symposium on Discrete Algorithms (SODA).*


A Affine maximizers are incentive compatible

Theorem A.1. For any choice of fixed weights and boosts, affine maximizers are incentive compatible.

Proof. (Following the standard structure for truthfulness of VCG in the static case.) Consider player \( i \) with true reward function \( r_i \) and reported rewards \( \tilde{r}_i \), where other players have rewards \( r_{-i} \). Recall that the payments charged for each agent can be written as

\[
p_i(r) = \frac{1}{w_i} \left( \text{asw}^{-i}(w, b, r) - \left( \mathbb{E}_{\pi^*} \left( \sum_{t=0}^{T} \left( w_j r_j(s_t, a_t) + b(s_t, a_t) \right) \right) \right) \right)
\]

where \( \text{asw}^{-i}(w, b, r) \) and reported rewards \( \tilde{r}_i \). The true expected utility at the time they report, as a function of their true reward function and a possible misreport \( \tilde{r}_i \), is

\[
U_i(r_i, \tilde{r}_i) = \mathbb{E}_{\pi^*} \left( \sum_{t=0}^{T} r_i(s_t, a_t) - \frac{1}{w_i} \left( \text{asw}^{-i}(w, b, r) - \left( \sum_{t=0}^{T} \left( w_j r_j(s_t, a_t) + b(s_t, a_t) \right) \right) \right) \right)
\]

where the expectation is over the randomness in the MDP (note that it does not need to be over the randomness in opponent types – IC should hold in dominant strategies considering the opponents).

In choosing a misreport \( \tilde{r}_i \), player \( i \) thus faces a maximization problem:

\[
\arg\max_{\tilde{r}_i} U_i(r_i, \tilde{r}_i) = \arg\max_{\tilde{r}_i} \mathbb{E}_{\pi^*} \left( \sum_{t=0}^{T} r_i(s_t, a_t) - \frac{1}{w_i} \left( \text{asw}^{-i}(w, b, r) - \left( \sum_{t=0}^{T} \left( w_j r_j(s_t, a_t) + b(s_t, a_t) \right) \right) \right) \right)
\]

\[
= \arg\max_{\tilde{r}_i} \mathbb{E}_{\pi^*} \left( \sum_{t=0}^{T} r_i(s_t, a_t) + \frac{1}{w_i} \left( \sum_{t=0}^{T} \sum_{j \neq i} w_j r_j(s_t, a_t) + b(s_t, a_t) \right) \right) - c
\]

\[
= \arg\max_{\tilde{r}_i} \mathbb{E}_{\pi^*} \left( \sum_{t=0}^{T} w_j r_j(s_t, a_t) + \sum_{t=0}^{T} \sum_{j \neq i} w_j r_j(s_t, a_t) + b(s_t, a_t) \right)
\]

where \( c \) is a constant. This is exactly the objective that the mechanism attempts to maximize if the bidder reports truthfully, so the best choice they can make is to do so.

IR and IC guarantees

In mechanism design, there is often a distinction between:

- \textit{Ex post} – properties that hold after all types have been reported and decisions have been made
- \textit{Ex interim} – properties that hold after a bidder has observed their own type, but before seeing other types.
- \textit{Ex ante} – properties that hold before types have been observed.

As far as the prior distribution of agent types is concerned, all our guarantees are \textit{ex post} – i.e. we ensure dominant-strategy incentive compatibility and ex-post IR.

In our problem, we have an additional source of randomness, the inherent randomness in the MDP itself. We thus refer to “in expectation” to refer to properties that hold when averaging over the randomness in the MDP, but ex post in the types.

Given the above choices of terminology, and given a correct policy and value estimate, our chosen mechanism and payment rules can guarantee in-expectation incentive compatibility and individual rationality.
B  Proof of Theorem 4.1

First, let us note that by Rademacher’s theorem differentiability almost everywhere follows from local Lipschitz continuity of a function. Moreover, the expected value and derivative are linear operators. Together this implies we only need to show that the expected values of $aw$ and $g_t$ are locally Lipschitz continuous.

In this proof, we identify boosts and agents’ rewards with vectors in $\mathbb{R}^{|S| \times |A|}$ and weights with vectors in $\mathbb{R}^n$. Moreover, we will assume that $f$ is a continuous density function, according to which $r$ is distributed and that it is sufficiently well-behaved, such that $\mathbb{E}_{\pi} [||r||_\infty]$ exists.

**Affine social welfare is Locally Lipschitz continuous**

Here we only show continuity in $w$, since the proof for $b$ works analogous. Fix $r \in \mathbb{R}^{|S| \times |A| \times n}$ and let $w, \hat{w} \in \mathbb{R}^n$, s.t. $\exists j : w_j \neq \hat{w}_j$. We need to show that $\exists K_r, \pi_j \in \mathbb{R}^n$, s.t.

$$
\pi = \pi^*(w, b, r) \quad \text{and} \quad \hat{\pi} = \pi^*(\hat{w}, b, r).
$$

To perform our proof we need the following inequalities:

- **By optimality of $\pi$**
  \[
  \sum_{i=1}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \pi} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \pi} [b(s_t, a_t)] 
  \geq \sum_{i=1, i \neq j}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \pi} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \pi} [b(s_t, a_t)]
  \]

- **By optimality of $\hat{\pi}$**
  \[
  \sum_{i=1}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [b(s_t, a_t)] 
  \leq \sum_{i=1, i \neq j}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [b(s_t, a_t)]
  \]

- **By assumption on $w, \hat{w}_j$**
  \[
  \sum_{i=1}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \pi} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \pi} [b(s_t, a_t)] 
  - \left( \sum_{i=1}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \pi} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \pi} [b(s_t, a_t)] \right) 
  \leq |w_j - \hat{w}_j| \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \pi} [r_j(s_t, a_t)]
  \]

- **By assumption on $w, \hat{w}_j$**
  \[
  \sum_{i=1}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [b(s_t, a_t)] 
  - \left( \sum_{i=1}^n \sum_{t=0}^T w_i \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [r_i(s_t, a_t)] + \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [b(s_t, a_t)] \right) 
  \leq |w_j - \hat{w}_j| \sum_{t=0}^T \mathbb{E}_{s_t, a_t \sim \hat{\pi}} [r_j(s_t, a_t)]
  \]
Now we have all the ingredients to show local Lipschitz continuity wrt to $w_j$. Let $K_r = \|r\|_{\infty} T \geq \max(\sum_{t=0}^{T} E_{s,t,a_i \sim \pi}[r_j(s_t, a_t)], \sum_{t=0}^{T} E_{s,t,a_i \sim \pi}[\hat{r}_j(s_t, a_t)])$. Then we have

$$
\sum_{i=1,i \neq j}^{n} \sum_{t=0}^{T} w_i E_{s,t,a_i \sim \pi}[r_j(s_t, a_t)] + \sum_{t=0}^{T} w_j E_{s,t,a_i \sim \pi}[\hat{r}_j(s_t, a_t)] + \sum_{t=0}^{T} E_{s,t,a_i \sim \pi}[b(s_t, a_t)] + K_r |w_j - \hat{w}_j|
$$

$$
\geq E_r(4) \sum_{i=1,i \neq j}^{n} \sum_{t=0}^{T} w_i E_{s,t,a_i \sim \pi}[r_j(s_t, a_t)] + \sum_{t=0}^{T} w_j E_{s,t,a_i \sim \pi}[\hat{r}_j(s_t, a_t)] + \sum_{t=0}^{T} E_{s,t,a_i \sim \pi}[b(s_t, a_t)] + K_r |w_j - \hat{w}_j|
$$

$$
\geq E_r(7) \sum_{i=1,i \neq j}^{n} \sum_{t=0}^{T} w_i E_{s,t,a_i \sim \pi}[r_j(s_t, a_t)] + \sum_{t=0}^{T} w_j E_{s,t,a_i \sim \pi}[\hat{r}_j(s_t, a_t)] + \sum_{t=0}^{T} E_{s,t,a_i \sim \pi}[b(s_t, a_t)]
$$

$$
\geq E_r(6) \sum_{i=1,i \neq j}^{n} \sum_{t=0}^{T} w_i E_{s,t,a_i \sim \pi}[r_j(s_t, a_t)] + \sum_{t=0}^{T} w_j E_{s,t,a_i \sim \pi}[\hat{r}_j(s_t, a_t)] + \sum_{t=0}^{T} E_{s,t,a_i \sim \pi}[b(s_t, a_t)] - K_r |w_j - \hat{w}_j|
$$

To conclude this proof, we note

$$
E_r[asw(w, b, r) - asw(\hat{w}, b, r)] \leq T E_r[\|r\|_{\infty}] |w_j - \hat{w}_j|
$$

where $E_r[\|r\|_{\infty}]$ is finite by assumption.

$g_k$ is locally Lipschitz continuous

Unlike for asw, we cannot argue that $g_k$ is Lipschitz continuous. Indeed changing the weights and boosts only slightly can cause a completely different policy to become optimal, leading to a discontinuous jump or drop in $\pi^*(w, b, r)$ and thus in general also in $g_k$. However, as we will show, when taking the expected value, these discontinuities get smoothed out, guaranteeing local Lipschitz continuity and thereby differentiability almost surely.

We will restrict to proving local Lipschitz continuity of $E_r[g_k(\pi^*(w, b, r), r)]$ with respect to $b$, as the proof with respect to $w$ works similar.

In the proofs below we identify the reported reward functions of all agents with vectors $r \in \mathbb{R}^d$, where $d = |S| \times |A| \times n$.

Our first observation is that all vectors $r$ which two policies $\pi_1, \pi_2$ give the same affine social welfare lie on a hyperplane. Indeed, let $\nu_i$ denote the induced state-action occupancy measure of policy $\pi_i$.

$$
\sum_{s,a} \nu_1(s,a)r_i(s,a) + b(s,a) = \sum_{s,a} \nu_2(s,a)r_i(s,a) + b(s,a)
$$

is equivalent to

$$
\sum_{s,a,i} (\nu_1(s,a) - \nu_2(s,a)) r_i(s,a) = \sum_{s,a} (\nu_2(s,a) - \nu_1(s,a))(b(s,a))
$$

Let $\nu = (w_1 \nu_1(s_1, a_1), w_2 \nu_1(s_1, a_1), \ldots, w_n \nu_1(s_1, a_1), w_1 \nu_1(s_2, a_1), \ldots, w_n \nu_1(s_1, a_1)).$ Then the above is equivalent to the following hyperplane:

$$
\mathcal{H}_{12}(w, b) = \{r : (\nu_1 - \nu_2)^T r = \sum_{s,a} (\nu_2(s,a) - \nu_1(s,a))(b(s,a))\}
$$

Let $b, \tilde{b} \in \mathbb{R}^{|S| \times |A|}$, s.t. $\exists! (s', a') : b(s', a') \neq \tilde{b}(s', a')$. $\tilde{b}$ gives us another hyperplane of equivalence between $\pi_1, \pi_2$

$$
\mathcal{H}_{12}(w, \tilde{b}) = \{r : (\nu_1 - \nu_2)^T r = \sum_{s,a} (\nu_2(s,a) - \nu_1(s,a))(\tilde{b}(s,a))\}
$$

Note that $\mathcal{H}_{12}(w, \tilde{b})$ and $\mathcal{H}_{12}(w, b)$ are parallel with a distance $\frac{|(\nu_2(s', a') - \nu_1(s', a'))(b(s', a') - \tilde{b}(s', a'))|}{\|\nu_1 - \nu_2\|}$. Moreover, for any given parameters $(w, b)$ there are only $|\Pi|^2$ planes of equivalence, where $|\Pi|$ is the number of deterministic policies.

When we change $b(s', a')$ to $\tilde{b}(s', a')$, then there can be rewards $r$, for which $\pi^*(w, b, r) \neq \pi^*(w, \tilde{b}, r)$. For these rewards it follows that in general $g_k(\pi^*(w, b, r), r)$ is different from $g_k(\pi^*(w, \tilde{b}, r), r)$. Using the hyperplanes defined above, we know the set of all $r$, where $g_k(\pi^*(w, b, r), r) \neq g_k(\pi^*(w, \tilde{b}, r), r)$ is contained in the set
Gradients can be computed in a straightforward manner.

For our second approach where we take the gradient through the regularised LP, we note that for the states containing the current timestep such that this simplifies to

\( |(\nu_j(s', a') - \nu_i(s', a'))(\tilde{b}(s', a') - b(s', a'))| \)

are the polytopes induced by the hyperplanes \( \mathcal{H}_{ij}(w, \tilde{b}) \) and \( \mathcal{H}_{ij}(w, b) \).

With this in mind let us make a first naive analysis of the difference of the change in expectation of \( g_k \), when changing \( b \).

\[
U(w, b, \tilde{b}) = \bigcup_{\pi, \pi' \in \Pi} U(w, b, \tilde{b})_{ij}
\]

where

\[
U(w, b, \tilde{b})_{ij} = \{ r : (\nu_i - \nu_j)^T r - \sum_{s, a} (\nu_j(s, a) - \nu_i(s, a))(b(s, a)) \leq (\nu_j(s', a') - \nu_i(s', a'))(\tilde{b}(s', a') - b(s', a')) \}
\]

for some constant \( C \) (since \( g_k = \mathcal{O}(\|r\|_\infty) \)).

We want to show that the above can be bounded by \( L|b(s', a') - \tilde{b}(s', a')| \) for some \( L \). For this we need to get a better understanding of \( \int_{U(w, b, \tilde{b})_{ij}} \|r\|_\infty f(r) dr \).

For the sake of simplicity, we assume now the probability density \( f \) has compact support on \( R^d \), i.e. there exists a \( K \) such that \( \forall r : \|r\|_\infty \leq K \implies f(r) = 0 \). Since the hyperplanes \( \mathcal{H}_{ij}(w, \tilde{b}) \) and \( \mathcal{H}_{ij}(w, b) \) have dimension \( d - 1 \) and are parallel with distance

\[
\frac{|(\nu_j(s', a') - \nu_i(s', a'))(b(s', a') - \tilde{b}(s', a'))|}{\|(\nu_1 - \nu_2)\|}
\]

we can bound the integral by multiplying an upper bound of the volume of \( U(w, b, \tilde{b})_{ij} \) with the maximum possible value of \( \|r\|_\infty f(r) \).

\[
\int_{U(w, b, \tilde{b})_{ij}} \|r\|_\infty f(r) dr \leq (2K)^{d-1}\frac{|(\nu_j(s', a') - \nu_i(s', a'))(b(s', a') - \tilde{b}(s', a'))|}{\|(\nu_1 - \nu_2)\|} K \max_r f(r)
\]

for \( L = (2K)^d \frac{|(\nu_j(s', a') - \nu_i(s', a'))|}{\|(\nu_1 - \nu_2)\|} \max_r f(r) \), which proves Lipschitz continuity and thereby differentiability almost surely.

### C Gradients of affine social welfare

As outlined in Section 4, if we can take the derivative inside the expected value, the implicit function theorem yields:

\[
\nabla_{w, b} \mathbb{E}_\rho [\mathcal{L}(\pi^*, w, b)] = \mathbb{E}_\rho [\nabla_2 \mathcal{L}(\pi^*, w, b) + \nabla_{w, b} \pi^*(w, b) \nabla_1 \mathcal{L}(\pi^*, w, b)]
\]

Here we show that we can analytically compute the partial derivatives with respect to \( w \) and \( b \) (keeping a policy \( \pi^* \) fixed) of \( \text{asw}, \text{sw} \) (and thus also for revenue), as well as for makespan, which is important given that we can use these to accelerate the gradient computation for our second approach where we take the gradient through the regularised LP. We note that for \( \text{asw}(w, b, \pi, r) \) the gradients can be computed in a straightforward manner.\(^9\) For this, rewrite \( \text{asw} \) using the state-action occupancy measure \( \nu_{\pi} \).\(^{10}\)

\[
\text{asw}(\pi, w, b, r) = \sum_{s, a} \nu_{\pi}(s, a) (\sum_{i=1}^n w_i r_i(s, a) + b(s, a))
\]

\(^9\)This assumption is not necessary. As long as \( f \) decays sufficiently quickly for large \( r \), the proof still goes through with some minor adjustments. However, we make the assumption here for streamlining our exposition and highlighting the parts of our proof, which are non-standard.

\(^{10}\)The analysis holds equivalently for \( \text{asw}_{\pi_h}(w, b, \pi_h) \)

In general finite horizon MDPs this would be defined as \( \nu_{\pi}(s, a) = \sum_{t=1}^T \mathbb{P}_\pi(s_t = s, a_t = a) \). In our experiments we always assume the states contain the current timestep such that this simplifies to \( \nu_{\pi}(s, a) = \mathbb{P}_\pi(s_t = s, a_t = a) \). Note that the same analysis equivalently holds for \( \text{asw}_{\pi_{\pi_h}} \).
In this form taking the partial derivative is straightforward. We get the following

\[ \nabla_w, \nabla_b \alpha w = \nabla_w, \nabla_b X \nabla \pi(s, a) (\sum_{i=1}^{n} w_i r_i(s, a) + b(s, a)) \]

\[ \nabla_w, \nabla_b \alpha w = \sum_{s, a} \pi(s, a) r_i(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{T} r_i(s, a) \right] \]

\[ \nabla_{b(s,a)} \alpha w = \pi(s, a) \]

Boosting a state increases \( \alpha w \) in relation to how often the state is visited under the optimal policy. This is easy to compute. In fact, \( \pi \) is the solution to the linear programming formulation of the MDP. Similarly increasing the weight of an agent changes \( \alpha w \) in proportion to the expected sum of rewards the agent gets.

For \( sw \) and makespan the analysis is even simpler, since the partial derivative with respect to \( w, b \), keeping a policy fixed is just 0. As both only depend on these parameters indirectly through the policy.

### D Proof of Theorem 4.3

**Proof.** Fix \( w, b, r \) and denote by \( L_{\alpha}(w, b, r), L(w, b, r) \) the optimal regularised and unregularised loss for this specific choice of variables. We first show

\[ \lim_{\alpha \to 0} L_{\alpha}(w, b, r) = L(w, b, r) \]

Let \( \pi^*, \pi^\alpha \) be the corresponding state-action measures to \( \pi^* \), \( \pi^\alpha \)—the optimal policies for the unregularised and regularised LP. Using Corollary 9 of Weed (2018), for sufficiently small \( \alpha \), we get:

\[ |L(w, b, r) - L_{\alpha}(w, b, r)| \]

\[ = \sum_{s, a} (\pi^*(s, a) - \pi^\alpha(s, a)) (q) \]

\[ = |\langle \pi^* - \pi^\alpha, q \rangle| \]

\[ \leq \|\pi^* - \pi^\alpha\|_1 \|q\|_{\infty} \]

\[ \leq 2R_1 \exp \left( - \frac{\Delta(r)}{\alpha R_1} + \frac{R_1 + R_H}{R_1} \right) (C\|r\|_{\infty}) \]

where \( C \) is some constant, \( R_1 \) is the \( l_1 \) radius of all feasible solutions, \( R_H \) is the entropic radius, and \( \Delta \) the suboptimality gap (Weed 2018). This shows pointwise convergence of \( L_{\alpha}(w, b, r) \). Note further that for any policy \( \pi \) and any \( w, b, \mathcal{L}_\pi(w, b, r) \) can be bounded by \( (C\|r\|_{\infty}) \) for some \( C \), which by assumption is integrable. It follows by the Dominated Convergence Theorem that

\[ \forall w, b : \lim_{\alpha \to 0} \mathbb{E} [L_{\alpha}(w, b, r)] = \mathbb{E} [L(w, b, r)] \]

### E Linear Program of regularized MDP

Below we give the regularized form of the MDP linear program – it is now a convex (exponential cone) program.

\[
\max \sum_{s \in S, a \in A} \left( \sum_{i=1}^{n} w_i r_i(s, a) + b(s, a) \right) \nu(s, a) + \alpha \sum_{s \in S, a \in A} \nu(s, a) \log \nu(s, a) \\
\text{s.t.} \sum_{a \in A} \nu(s, a) = \sum_{s', a'} P(s|s', a') \nu(s', a') + \mu_0(s) \quad \forall s \in S \\
\nu(s, a) \geq 0 \quad \forall s \in S, a \in A
\]
The grid search experiments and all experiments in the gridworld environment were run concurrently on a server with 256 cores and 250GB of RAM, while restricting the number of threads in MOSEK to 4. Other experiments were run on cluster nodes with 4 cores and 1GB or 2GB of RAM per core, except that the task scheduling regularized LP jobs with 3 agents were run with 16 cores and 64GB memory. During development we experimented with up to 1000 sampled valuation profiles, up to 2000 perturbations, learning rates ranging from 0.001 to 0.1, and regularization strengths up to 0.1; we quickly settled on the chosen hyperparameters and did not do a more exhaustive search due to computational constraints. For distributions where the bidder valuations are symmetric, we optimize only boosts, fixing the weights to 1.