An Online-Learning Approach to Inverse Optimization

Andreas Bärmann[1], Alexander Martin[1], Sebastian Pokutta[2] and Oskar Schneider[3]

[1] Andreas.Baermann@math.uni-erlangen.de
Alexander.Martin@math.uni-erlangen.de
Lehrstuhl für Wirtschaftsmathematik,
Department Mathematik,
Friedrich-Alexander-Universität Erlangen-Nürnberg,
Cauerstraße 11, 91058 Erlangen, Germany

[2] Sebastian.Pokutta@isye.gatech.edu
Laboratory for Interactive Optimization and Learning,
School of Industrial & Systems Engineering,
Georgia Institute of Technology,
Ferst Drive 755, NW, Atlanta, Georgia, USA

[3] Oskar.Schneider@fau.de
Gruppe Data Science and Optimization
Fraunhofer Arbeitsgruppe für Supply-Chain Services SCS,
Fraunhofer Institut für Integrierte Schaltungen IIS,
Nordostpark 93, 90411 Nürnberg, Germany

30th October 2018

Abstract

In this paper, we demonstrate how to learn the objective function of a decision-maker while only observing the problem input data and the decision-maker's corresponding decisions over multiple rounds. Our approach is based on online learning and works for linear objectives over arbitrary feasible sets for which we have a linear optimization oracle. As such, it generalizes previous approaches based on KKT-system decomposition and dualization. The two exact algorithms we present – based on multiplicative weights updates and online gradient descent respectively – converge at a rate of $\mathcal{O}(1/\sqrt{T})$ and thus allow taking decisions which are essentially as good as those of the observed decision-maker already after relatively few observations. We also discuss several useful generalizations, such as the approximate learning of non-linear objective functions and the case of suboptimal observations. Finally, we show the effectiveness and possible applications of our methods in a broad computational study.

Keywords: Learning Objective Functions, Online Learning, Multiplicative Weights Update Algorithm, Online Gradient Descent, Mixed-Integer Programming

Mathematics Subject Classification: 68Q32 - 68T05 - 90C90 - 90C27 - 90C11

1 Introduction

Human decision-makers are very good at taking decisions under rather imprecise specification of the decision-making problem, both in terms of constraints as well as objective. One

might argue that the human decision-maker can pretty reliably learn from observed previous decisions – a traditional learning-by-example setup. At the same time, when we try to turn these decision-making problems into actual optimization problems, we often run into all types of issues in terms of specifying the model. In an optimal world, we would be able to *infer or learn* the optimization problem from previously observed decisions taken by an expert.

This problem naturally occurs in many settings where we do not have direct access to the decision-maker's preference or objective function but can observe his behaviour, and where the learner as well as the decision-maker have access to the same information. Natural examples are as diverse as making recommendations based on user history and strategic planning problems, where the agent's preferences are unknown but the system is observable. Other examples include knowledge transfer from a human planner into a decision support system: often human operators have arrived at finely-tuned "objective functions" through many years of experience, and in many cases it is desirable to replicate the decision-making process both for scaling up and also for potentially including it in large-scale scenario analysis and simulation to explore responses under varying conditions.

Here we consider the learning of preferences or objectives from an expert by means of observing his actions. More precisely, we observe a set of input parameters and corresponding decisions of the form $\{(p_1, x_1), \ldots, (p_T, x_T)\}$. They are such that $p_t \in P$ with $t = 1, \ldots, T$ is a certain realization of problem parameters from a given set $P \subseteq \mathbb{R}^k$ and x_t is an optimal solution to the optimization problem

$$\begin{aligned} \max \quad & c_{\text{true}}^\top x \\ \text{s.t.} \quad & x \in X(p_t), \end{aligned}$$

where $c_{\text{true}} \in \mathbb{R}^n$ is the expert's true but unknown objective and $X(p_t) \subseteq \mathbb{R}^n$ for some (fixed) n. We assume that we have full information on the feasible set $X(p_t)$ and that we can compute $\operatorname{argmax}\{c^\top x \mid x \in X(p_t)\}$ for any candidate objective $c \in \mathbb{R}^n$ and $t = 1, \ldots, T$. We present two online-learning algorithms, based on the multiplicative weights update (MWU) algorithm and online gradient descent (OGD) respectively, that allow us to learn a strategy $(c_1, \ldots, c_T)$ of subsequent objective function choices with the following guarantee: if we optimize according to the surrogate objective function c_t instead of the actual unknown objective function c_{true} in response to parameter realization p_t, we obtain a sequence of optimal decisions (w.r.t. to each c_t) given by

$$\bar{x}_t = \operatorname{argmax}\{c_t^\top x \mid x \in X(p_t)\}$$

that are essentially as good as the decisions x_t taken by the expert on average. To this end, we interpret the observations of parameters and expert solutions as revealed over multiple rounds such that in each round t we are shown the parameters p_t first, then take our optimal decision $\bar{x}_t$ according to our objective function c_t, then we are shown the solution x_t chosen by the expert, and finally we are allowed to update c_t for the next round. For this setup, we will be able to show that our MWU-based algorithm attains an error bound of

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{\text{true}})^\top (\bar{x}_t - x_t) \leq 2K \sqrt{\frac{\ln n}{T}},$$

where $K \geq 0$ is an upper bound on the ℓ_∞-diameter of the feasible regions $X(p_t)$ with $t = 1, \ldots, T$. This implies that both the deviations in true cost $c_{\text{true}}^\top (x_t - \bar{x}_t) \geq 0$ as well as the deviations in surrogate cost $c_t^\top (\bar{x}_t - x_t) \geq 0$ can be made arbitrarily small on average. In other words, the average regret for having decided optimally according to the surrogate objectives c_t vs. having decided optimally for the true objective c_{true} vanishes at a rate of $\mathcal{O}(1/\sqrt{T})$. While this algorithm is only applicable if $c_{\text{true}} \geq 0$ holds, our algorithm based on OGD works without this restriction. If $K \geq 0$ is an upper bound on the ℓ_2-diameter of the feasible regions $X(p_t)$,

$t = 1, \ldots, T$ and $L \geq 0$ is an upper bound on the ℓ_2-diameter of the set F from which both c_{true} and the c_t's originate, then the OGD-based algorithm achieves an error bound of

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{\text{true}})^\top (\bar{x}_t - x_t) \leq \frac{3LK}{2\sqrt{T}}.$$

These results show that linear objective functions over general feasible sets can be learned from relatively few observations of historical optimal parameter-solutions pairs. We will derive various extensions of our scheme, such as approximately learning non-linear objective functions and learning from suboptimal decisions. We will also, briefly, discuss the case where the objective c_{true} is known, but some linear constraints are unknown in this paper.

Literature Overview

The idea of learning or inferring parts of an optimization model from data is a reasonably well-studied problem under many different assumptions and applications and has gained significant attention in the optimization community over the last few years, as discussed for example in den Hertog and Postek (2016), Lodi (2016) or Simchi-Levi (2014). These papers argue that there would be significant benefits in combining traditional optimization models with data-derived components. Most approaches in the literature focus on deriving the objective function of an expert decision-maker in a static fashion, based on past observations of input data and the decisions he took in each instance. In almost all cases, the objective functions are learned by considering the KKT-conditions or the dual of the (parameterized) optimization problem, and as such convexity for both the feasible region and the objective function is inherently assumed. Examples of this approach include Keshavarz et al. (2011), Li (2016) as well as Thai and Bayen (2018), where the latter one also considers the derivation of variational inequalities from data. Sometimes also distributional assumptions regarding the observations are made. Applications of such approaches have been heavily studied in the context of energy systems (Ratliff et al. (2014); Konstantakopoulos et al. (2016)), robot motion (Papadopoulos et al. (2016); Yang et al. (2014)), medicine (Sayre and Ruan (2014)) and revenue management (Kallus and Udell (2015); Qiang and Bayati (2016); Chen et al. (2015); Kallus and Udell (2016); Bertsimas and Kallus (2016)); also in the situation where the observed decisions were not necessarily optimal (Nielsen and Jensen (2004)).

Very closely related to our learning approach in terms of the problem formulation is Esfahani et al. (2018). This work studies different loss functions for evaluating a learned objective function on a data sample (p_t, x_t), which leads the authors to the minimization of the same regret function that we consider in the present paper. However, as their solution approach is based on duality, it does not extend to the integer case like the ideas presented here. Also closely related is the research reported in Troutt et al. (2005), which was later extended in Troutt et al. (2006), where an optimization model is defined that searches for a linear optimization problem that minimizes the total difference between the observed solutions and solutions found by optimizing according to that optimization problem. In the latter case, the models are solved using LP duality and cutting planes. In the follow-up work Troutt et al. (2008), a genetic algorithm is used to solve the problem heuristically under rather general assumptions, but inherently without any quality guarantees, and in Troutt et al. (2011) the authors study experimental setups for learning objectives under various stochastic assumptions, focussing on maximum likelihood estimation, which is generally the case for their line of work; we make no such assumptions.

Closely related to learning optimization models from observed data is the subject of inverse optimization. Here the goal is to find an objective function that renders the observed solutions optimal with respect to the concurrently observed parameter realizations. Approaches in this

field mostly stem from convex optimization, and they are used for inverse optimal control (Iyengar and Kang (2005); Panchea and Ramdani (2015); Molloy et al. (2016)), inverse combinatorial optimization (D. Burton (1997); Burton and Toint (1994, 1992); Sokkalingam et al. (1999); Ahuja and Orlin (2000)), integer inverse optimization (Schaefer (2009)) and inverse optimization in the presence of noisy data, such as observed decisions that were suboptimal (Aswani et al. (2018); Chan et al. (2018)).

All these approaches heavily rely on duality and thus require convexity assumptions both for the feasible region as well as the objectives. As such, they cannot deal with more complex, possibly non-convex decision domains. This in particular includes the important case of integer-valued decisions (such as yes/no-decisions or, more generally, mixed-integer programming) and also many other non-convex setups (several of which admit efficient linear optimization algorithms). Previously, this was only possible when the structure of the feasible set could be beneficially exploited. In contrast, our approach does not make any such assumptions and only requires access to a *linear optimization oracle* (in short: *LP oracle*) for the feasible region X. Such an oracle is defined as a method which, given a vector $c \in \mathbb{R}^n$, returns $\operatorname{argmax}\{c^\top x \mid x \in X\}$.

Also related to our work is inverse reinforcement learning and apprenticeship learning, where the reward function is the target to be learned. However, in this case the underlying problem is modelled as a Markov decision process (MDP); see, for example, the results in Syed and Schapire (2007) and Ratia et al. (2012). Typically, the obtained guarantees are of a different form though. Similarly, our work is not to be confused with the methods developed in Taskar et al. (2005) and Daumé et al. (2005), where online algorithms are used for learning aggregation vectors for edge features in graphs, with inverse optimization as a subroutine to define the update rule. In contrast, we do inverse optimization by means of online-learning algorithms, which is basically the reverse setup.

Our approach is based on online learning, and we mainly use the simple *EXP* algorithm here to attain the stated asymptotic regret bound. The EXP algorithm is commonly also called *Multiplicative Weights Update (MWU)* algorithm and was developed in Littlestone and Warmuth (1994), Vovk (1990) as well as Freund and Schapire (1997) (see Arora et al. (2012); Hazan (2016) for a comprehensive introduction; see also Audibert et al. (2013)). A similar algorithm was used in Plotkin et al. (1995) for solving fractional packing and covering problems. To generalize the applicability of our approach, we also derive a second algorithm based on *Online Gradient Descent (OGD)* due to Zinkevich (see Zinkevich (2003)). We finally point out that our feedback is stronger than bandit feedback. This requirement is not unexpected as the costs chosen by the "adversary" depend on our decision; as such the bandit model (see, for example, Dani et al. (2008), Abbasi-yadkori et al. (2011)) does not readily apply.

Contribution

To the best of the authors' knowledge, this paper makes the first attempt to learn the objective function of an optimization model from data using an online-learning approach.

Online Learning of Optimization Problems Based on samples for the input-output relationship of an optimization problem solved by a decision-maker, our aim is to learn an objective function which is consistent with the observed input-output relationship. This is indeed the best one can hope for: an adversary could play the same environment for T rounds and then switch. This is less of an issue if the environments form samples that are independent and identically distributed (i.i.d.) from some distribution.

In our setup, the expert solves the decision-making problem repeatedly for different input parameter realizations. From these observations, we are able to learn a strategy of objective

functions that emulate the expert's unknown objective function such that the difference in solution quality between the solutions converges to zero on average.

While previous methods based on dualization or KKT-system-based approaches can lead to similar or even stronger results in the continuous/convex case, online learning allows us to relax this convexity requirement and to work with arbitrary decision domains as long as we are able to optimize a linear function over them, in particular mixed-integer programs (MIPs). Thus, we do not explicitly analyze the KKT-system or the dual program (in the case of linear programs (LPs); see Remark 3.1). In particular, one might consider our approach as an algorithmic analogue of the KKT-system (or dual program) in the convex case.

To summarize, we stress that (a) we do not make any assumptions regarding distribution of the observations, (b) the observations can be chosen by a fully-adaptive adversary, and (c) we do not require any convexity assumptions regarding the feasible regions and only rely on access to an LP oracle. We would also like to mention that our approach can be extended to work with slowly changing objectives using appropriate online-learning algorithms such as, for example, those found in Jadbabaie et al. (2015) or Zinkevich (2003); the regret bounds will depend on the rate of change.

A Broad Computational Study We conduct extensive experiments to demonstrate the effectiveness and wide applicability of our algorithmic approach. To this end, we investigate its use for learning the objective functions of several combinatorial optimization problems that frequently occur in practice (possibly as subproblems of larger problems) and explore, among other things, how well the learned objective generalizes to unseen data samples.

The present paper is the full version of an extended abstract submitted to the International Conference on Machine Learning (ICML) 2017, see Bärmann et al. (2017).

2 Problem Setting

We consider the following family of optimization problems $(\mathrm{OPT}(p))_p$, which depend on parameters $p \in P \subseteq \mathbb{R}^k$ for some $k \in \mathbb{N}$:

$$\begin{aligned} \max \quad & c_{\text{true}}^\top x \\ \text{s.t.} \quad & x \in X(p), \end{aligned}$$

where $c_{\text{true}} \in \mathbb{R}^n$ is the objective function and $X(p) \subseteq \mathbb{R}^n$ is the feasible region, which depends on the parameters p. Of particular interest to us will be feasible regions that arise as polyhedra defined by linear constraints and their intersections with integer lattices, i.e. the cases of LPs and MIPs:

$$X(p) = \{x \in \mathbb{Z}^{n-l} \times \mathbb{R}^l \mid A(p)x \leq b(p)\}$$

with $A(p) \in \mathbb{R}^{m \times n}$ and $b(p) \in \mathbb{R}^m$. However, our approach can also readily be applied in the case of more complex feasible regions, such as mixed-integer sets bounded by convex functions:

$$X(p) = \{x \in \mathbb{Z}^{n-l} \times \mathbb{R}^l \mid G(p, x) \leq 0\}$$

with $G\colon P \times \mathbb{Z}^{n-l} \times \mathbb{R}^l \to \mathbb{R}$ convex – or even more general settings. In fact, for any possible choice of model for the sets of feasible decisions, we only require the availability of a linear optimization oracle, i.e. an algorithm which is able to determine $\operatorname{argmax}\{c^\top x \mid x \in X(p)\}$ for any $c \in \mathbb{R}^n$ and $p \in P$. We call a decision $x \in \mathbb{R}^n$ *optimal* for p if it is an optimal solution to $\mathrm{OPT}(p)$.

We assume that Problem $\mathrm{OPT}(p)$ models a parameterized optimization problem which has to be solved repeatedly for various input parameter realizations p. Our task is to learn the fixed objective function c_{true} from given observations of the parameters p and a corresponding

optimal solution x to $\mathrm{OPT}(p)$. To this end, we further assume that we are given a series of observations $((p_t, x_t))_t$ of parameter realizations $p_t \in P$ together with an optimal solution x_t to $\mathrm{OPT}(p_t)$ computed by the expert for $t = 1, \ldots, T$; these observations are revealed over time in an online fashion: in round t, we obtain a parameter setting p_t and compute an optimal solution $\bar{x}_t \in X(p_t)$ with respect to an objective function c_t based on what we have learned about c_{true} so far. Then we are shown the solution x_t the expert with knowledge of c_{true} would have taken and can use this information to update our inferred objective function for the next round. In the end, we would like to be able to use our inferred objective function to take decisions that are essentially as good as those chosen by the expert in an appropriate aggregation measure such as, for example, "on average" or "with high probability". The quality of the inferred objective is measured in terms of cost deviation between our solutions $\bar{x}_t$ and the solutions x_t obtained by the expert – details of which will be given in the next section.

To fix some useful notations, let $v(i)$ denote the i-th component of a vector v throughout, and let $[n] := \{1, \ldots, n\}$ for any natural number n. Furthermore, let $\mathbb{1}^n := (1, \ldots, 1)^\top$ denote the all-ones vector in $\mathbb{R}^n$. Finally, we need a suitable measure for the diameter of a given set.

Definition 2.1. *The ℓ_p-diameter of a set $S \subseteq \mathbb{R}^n$, denoted by $\mathrm{diam}_p(S)$, is the largest distance between any two points $x_1, x_2 \in S$, measured in the p-norm, $1 \leq p \leq \infty$, i.e.*

$$\mathrm{diam}_p(S) := \max_{x_1, x_2 \in S} \|x_1 - x_2\|_p.$$

As a technical assumption, we further demand that $c_{\text{true}} \in F$ for some convex, compact and non-empty subset $F \subseteq \mathbb{R}^n$, which is known beforehand. This is no actual restriction, as F could be chosen to be any ball according to some p-norm, $1 \leq p \leq \infty$, for example. In particular, this ensures that we do not have to deal with issues that arise when rescaling our objective.

3 Learning Objectives

Ideally, we would like to find the true objective function c_{true} as a solution to the following optimization problem:

$$\min_{c \in F} \sum_{t \in [T]} \left(\left(\max_{x \in X(p_t)} c^\top x \right) - c^\top x_t \right), \tag{1}$$

where $\|\cdot\| \colon \mathbb{R}^n \to \mathbb{R}_+$ is an arbitrary norm on $\mathbb{R}^n$ and $x_t \in X(p)$ is the *optimal* decision taken by the expert in round t. The true objective function c_{true} is an optimal solution to Problem (1) with objective value 0. This is because any solution $\hat{c} \in F$ is feasible and produces non-negative summands

$$\left(\max_{x \in X(p_t)} \hat{c}^\top x \right) - \hat{c}^\top x_t$$

for $t \in [T]$, as we assume $x_t \in X(p_t)$ to be optimal for p_t with respect to c_{true}.

Problem (1) contains T instances of the following maximization subproblem:

$$\max \quad c^\top x \tag{2a}$$
$$\text{s.t.} \quad x \in X(p_t). \tag{2b}$$

For each $t = 1, \ldots, T$, the corresponding Subproblem (2) asks for an optimal solution $\bar{x}_t$ when optimizing over the feasible set $X(p_t)$ with a given $c \in F$ as the objective function. When solving Problem (1), we are interested in an objective function vector $c \in F$ that delivers a consistent explanation for why the expert chose x_t as his response to the parameters p_t in round $t = 1, \ldots, T$. We call an objective function $c \in F$ from some prescribed set of objective functions $F \subseteq \mathbb{R}^n$ *consistent* with the observations (p_t, x_t), $t \in [T]$, if it is optimal for the resulting Problem (1). The aim is to find an objective $c \in F$ for which the optimal solution of Subproblem (2) attains

a value as close as possible to that of the expert's decision, averaged over all observations. The approaches we present here will provide even stronger guarantees in some cases, such as the one described in Section 3.2, showing that we can replicate the decision-making behaviour of the expert.

Remark 3.1. *Note that in the case of polyhedral feasible regions, i.e.* $p_t = (A_t, b_t) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ *and* $X(p_t) = \{x \in \mathbb{R}^n \mid A_t x \leq b_t\}$ *for* $t = 1, \ldots, T$*, as well as a polyhedral region* $F = \{c \in \mathbb{R}^n \mid Bc \leq d\}$*, Problem* (1) *can be reformulated as a linear program by dualizing the* T *instances of Subproblem* (2)*. This yields*

$$\begin{aligned}
\min \quad & \sum_{t=1}^{T} (b_t^\top y_t - c^\top x_t) && && \text{(3a)}\\
\text{s.t.} \quad & A_t^\top y_t &= c \quad & (\forall t = 1, \ldots, T) && \text{(3b)}\\
& y_t &\geq 0 \quad & (\forall t = 1, \ldots, T) && \text{(3c)}\\
& Bc &\leq d, && && \text{(3d)}
\end{aligned}$$

where the y_t *are the corresponding dual variables and the* x_t *are the observed decisions from the expert (i.e. the latter are part of the* input data*). This problem asks for a primal objective function vector* c *that minimizes the total duality gap summed over all primal-dual pairs* (x_t, y_t) *while all* y_t*'s shall be dual feasible, which makes the* x_t*'s the respective primal optimal solutions. Thus, Problem* (1) *can be seen as a direct generalization of the linear primal-dual optimization problem. In fact, our approach also covers non-convex cases, e.g. mixed-integer linear programs.*

Problem (1) can be interpreted as a game over T rounds between a player who chooses an objective function c_t in round $t \in [T]$ and a player who knows the true objective function c_{true} and chooses the observations (p_t, x_t) in a potentially adversarial way. The payoff of the latter player in each round t is equal to $c_t^\top (\bar{x}_t - x_t) \geq 0$, i.e. the difference in cost between our solution and the expert's solution as given by our guessed objective function c_t.

As Problem (1) is hard to solve in general, we will design online-learning algorithms that, rather than finding an optimal objective c, find a *strategy* of objective functions $(c_1, c_2, \ldots, c_T)$ to play in each round whose error in solution quality as compared to the true objective function is as small as possible. Our aim will then be to give a quality guarantee for this strategy in terms of the number of observations.

To allow for approximation guarantees, it will not only be necessary that the set of possible objective functions to choose from is bounded, but also that the observed feasible sets have a common upper bound on their ℓ_p-diameter.

From a meta perspective, our approach works as outlined in Algorithm 1. It chooses an

Algorithm 1 Online Objective Function Learning

Input: Observations (p_t, x_t) for $t = 1, \ldots, T$
Output: Sequence of objectives $(c_1, c_2, \ldots, c_T)$

```
Choose initial objective function c_1 ∈ F
for t = 1, ..., T do
    Observe parameters p_t
    Compute x̄ ∈ X(p_t) as an optimal solution to Subproblem (2) with objective c_t
    Observe expert solution x_t ∈ X(p_t)
    Compute an updated objective c_{t+1} ∈ F
end for
return (c_1, c_2, ..., c_T).
```

arbitrary objective in the first round, as there is no better indication of what to do at this point.

Then, in each round $t = 1, \dots, T$, it computes an optimal solution over $X(p_t)$ with respect to the current guess of objective function c_t. Upon the following observation of the expert's solution, it updates its guess of objective function to use it in the next round.

Clearly, the accumulated objective value of a strategy $(c_1, \dots, c_T)$ over T rounds is given by $\sum_{t=1}^{T} c^\top \bar{x}_t$, while that of c_{true} would be $\sum_{t=1}^{T} c_{\text{true}}^\top x_t$. Via the proposed scheme, it would be overly ambitious to demand $\lim_{T\to\infty} c_T = c_{\text{true}}$, or even $\lim_{T\to\infty} c_T^\top \bar{x}_T = c_{\text{true}}^\top x_T$ as the following example shows.

Example 3.2. *Consider the case $c_{true} = (0,1)^\top$ and $X(p_t) \subset \{x \in \mathbb{R}^2 \mid x(1) = 0\}$ for $t = 1, \dots, T$. If the first player chooses $c_t = (1-\varepsilon, \varepsilon)^\top$ for some $0 < \varepsilon \leq 1$ as his objective function guess in each round $t = 1, \dots, T$, he will obtain optimal solutions $\bar{x}_t$ with respect to c_{true}. However, both the objective functions c_t and the objective values $c_t^\top \bar{x}_t$ will be far off. Indeed, when taking the 1-norm, we have $\|c_{true} - c_t\|_1 = \|(\varepsilon - 1, 1 - \varepsilon)^\top\|_1 \to 2$ for $\varepsilon \to 0$. And if $X(p_t) = \{(0,1)^\top\}$ for all $t = 1, \dots, T$, we additionally have $c_{true}^\top (0,1)^\top = 1$, but $c_t^\top (0,1)^\top = \varepsilon \to 0$ for $\varepsilon \to 0$.*

Altogether, we cannot expect to approximate the true objective function c_{true} or the true optimal values $c_{\text{true}}^\top x_t$ in general. Neither can we expect to approximate the solutions x_t, because even if we have the correct objective function $c_t = c_{\text{true}}$ in each round, the optima do not not necessarily have to be unique.

As a more appropriate measure of quality, we will show that our algorithms based on online learning produce strategies $(c_1, \dots, c_T)$ with

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{\text{true}})^\top (\bar{x}_t - x_t) = 0, \tag{4}$$

of which we will see that it directly implies both

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} c_t^\top (\bar{x}_t - x_t) = 0 \tag{5}$$

and

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^\top (x_t - \bar{x}_t) = 0, \tag{6}$$

with non-negative summands for all rounds t in all three expressions. The *objective error* $\sum_{t=1}^{T} c_t^\top (\bar{x}_t - x_t)$ is the objective function of Problem (1) when relaxing the requirement to play the same objective function in each round and instead passing to a strategy of objective functions. Equation (5) states that the average objective error over all observations converges to zero with the number of observations going to infinity. The *solution error* $\sum_{t=1}^{T} c_{\text{true}}^\top (x_t - \bar{x}_t)$ is the cumulative suboptimality of the solutions $\bar{x}_t$ compared to the optimal solutions x_t with respect to the true objective function. According to Equation (6), it equally tends to zero on average with an increasing number of observations. This means it is possible to take decisions $\bar{x}_t$ which are essentially as good as the decisions x_t of the expert with respect to c_{true} over the long run.

Our measure of quality of a strategy of objective functions (4) is derived from the notion of *regret*, which is commonly used in online learning to characterize the quality of a learning algorithm: given an algorithm A which plays solutions $x_t \in K$ from some decision set F in response to loss functions $f_t \colon F \to R$ observed from an adversary over rounds $t = 1, \dots, T$, it is given by $R(A) \coloneqq \sum_{t=1}^{T} f_t(x_t) - \min_{x \in F} f_t(x)$. Minimizing the regret of a sequence of decisions thus aims to find a strategy that perfoms at least as good as the *best fixed decision in hindsight*, i.e. the best static solution that can be played with full advance-knowledge of the loss functions the adversary will play. See Hazan (2016), for example, for a broad introduction to regret minimization in online learning.

In our approach, we interpret the set of possible objective functions F in Problem (1) as the set of feasible decisions from which our learning algorithms choose an objective c_t in each

round $t \in [T]$. Furthermore, we use $(\bar{x}_t - x_t)$ as the corresponding loss function in round t. We are then interested in the regret against c_{true}, which is given by $\sum_{t=1}^{T}(c_t - c_{\text{true}})^\top(\bar{x}_t - x_t)$. Equation (4) states that the average of this *total error* tends to zero as the number of observations increases. Note that c_{true} is not necessarily the best fixed objective in hindsight – the latter would be given by a standard unit vector e_i, where $i \in \operatorname{argmin}\{i \in 1, \ldots, n \mid \sum_{t=1}^{T}(\bar{x}_t(i) - x_t(i))\}$, which is rather meaningless in our case.

In the following, we derive two online-learning algorithms for which Equation (4) holds provably as wells as an intuitive heuristic for LPs for which Equation (4) holds empirically in our experiments in Section 4.

3.1 An Algorithm based on Multiplicative Weights Updates

A classical algorithm in online learning is the multiplicative weights update (MWU) algorithm, which solves the following problem: given a set of n decisions, a player is required to choose one of these decisions in each round $t \in [T]$. Each time, after the player has chosen his decision, an adversary reveals to him the costs $m_t \in [-1, 1]^n$, of the decisions in the current round. The objective of the player is to minimize his overall cost $\sum_{t=1}^{T} m_t^\top w_t$ over the time horizon T. The MWU algorithm solves this problem by maintaining weights $w_t \in \mathbb{R}_+^n$ which are updated from round to round, starting with the initial weights $w_1 = \mathbb{1}^n$. These weights are used to derive a probability distribution $p_t := w_t / \|p_t\|$. In round t, the player samples a decision i from $\{1, \ldots, n\}$ according to p_t. Upon observation of the costs m_t, the player updates his weights according to

$$w_{t+1} = w_t - \eta(w_t \odot m_t),$$

where $0 < \eta < \frac{1}{2}$ is a suitable step size, in online learning also called *learning rate*, and $a \odot b := (a_1 \cdot b_1, \ldots, a_n \cdot b_n)$ denotes the componentwise multiplication of two vectors $a, b \in \mathbb{R}^n$. The expected cost of the player in round t is then given by $m_t^\top p_t$, and the total expected cost is given by $\sum_{t=1}^{T} m_t^\top p_t$. MWU attains the following regret bound against any fixed distribution:

Lemma 3.3 (Arora et al. (2012, Corollary 2.2))**.** *The MWU algorithm guarantees that after T rounds, for any distribution p on the decisions, we have*

$$\sum_{t=1}^{T} m_t^\top p_t \leq \sum_{t=1}^{T}(m_t + \eta|m_t|)^\top p + \frac{\ln n}{\eta},$$

where the $|m_t|$ is to be understood componentwise.

The above regret bound is valid for any distribution p, in particular for the best distribution p_{best} in hindsight, i.e. the distribution that would have performed best given the observed cost vectors m_t. The latter is again given by some standard unit vector.

We will now reinterpret the distributions p_t, a suitable distribution p_{true} to compare their regret to as well as the cost vectors m_t in MWU in a way that will allow us to learn an objective function from observed solutions. Namely, we will identify the distributions p_t with the objective functions c_t in the strategy of the player and the distribution p_{true} with the actual objective function c_{true}. The difference between the optimal solution $\bar{x}_t$ computed by the player and the optimal solution x_t of the expert will then act as the cost vector m_t (after appropriate normalization).

Naturally, this limits us to $F = \Delta_n := \{c \in \mathbb{R}_+^n \mid \|c\|_1 = 1\}$, i.e. the objective functions have to lie in the positive orthant (while normalization is without loss of generality). However, whenever this restriction applies, we obtain a very lightweight method for learning the objective function of an optimization problem. In Section 3.3, we will present an algorithm which works without this assumption on F.

Our application of MWU to learning the objective function of an optimization problem proceeds as outlined in Algorithm 2.

Algorithm 2 Objective Function Learning via Multiplicative Weights Updates

Input: Observations (p_t, x_t) for $t = 1, \ldots, T$
Output: Sequence of objectives $(c_1, c_2, \ldots, c_T)$

$\eta \leftarrow \sqrt{\frac{\ln n}{T}}$ {Set learning rate}
$w_1 \leftarrow \mathbb{1}^n$ {Initialize weights}
for $t = 1, \ldots, T$ **do**
  $c_t \leftarrow \frac{w_t}{\|w_t\|_1}$ {Normalize weights}
  Observe parameters p_t
  $\bar{x}_t \leftarrow \operatorname{argmax}\{c_t^\top x \mid X(p_t)\}$ {Solve Subproblem (2)}
  Observe expert solution x_t
  if $\bar{x}_t = x_t$ **then**
    $y_t \leftarrow 0$
  else
    $y_t \leftarrow \frac{\bar{x}_t - x_t}{\|\bar{x}_t - x_t\|_\infty}$
  end if
  $w_{t+1} \leftarrow w_t - \eta(w_t \odot y_t)$ {Update weights}
end for
return $(c_1, c_2, \ldots, c_T)$.

For the series of objectives functions $(c_t)_t$ that our algorithm returns, we can establish the following guarantee:

Theorem 3.4. *Let $K \geq 0$ with $\operatorname{diam}_\infty X(p_t) \leq K$ for all $t = 1, \ldots, T$. Then we have*

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{true})^\top (\bar{x}_t - x_t) \leq 2K\sqrt{\frac{\ln n}{T}},$$

and in particular it also holds:

1. $0 \leq \frac{1}{T} \sum_{t=1}^{T} c_t^\top (\bar{x}_t - x_t) \leq 2K\sqrt{\frac{\ln n}{T}}$,
2. $0 \leq \frac{1}{T} \sum_{t=1}^{T} c_{true}^\top (x_t - \bar{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}}$.

Proof. According to the standard performance guarantee of MWU from Lemma 3.3, Algorithm 2 attains the following bound on the total error of the secuence (c_t) compared to c_{true} with respect to the cost vectors y_t:

$$\sum_{t=1}^{T} c_t^\top y_t \leq \sum_{t=1}^{T} c_{\text{true}}^\top (y_t + \eta |y_t|) + \frac{\ln n}{\eta},$$

where the $|y_t|$ is to be understood component-wise. Using that each each entry of $|y_t|$ is at most 1 and dividing by T, we can conclude

$$\frac{1}{T} \sum_{t=1}^{T} c_t^\top y_t - \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^\top y_t \leq \eta \sum_{i=1}^{n} c_{\text{true}}(i) + \frac{\ln n}{\eta T}$$

and further, as $c_{\text{true}} \in F$,

$$\frac{1}{T} \sum_{t=1}^{T} c_t^\top y_t - \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^\top y_t = \eta + \frac{\ln n}{\eta T}.$$

The right-hand side attains its minimum for $\eta = \sqrt{\frac{\ln n}{T}}$, which yields the bound

$$\frac{1}{T} \sum_{t=1}^{T} c_t^\top y_t - \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^\top y_t \leq 2\sqrt{\frac{\ln n}{T}}.$$

Substituting back for the y_t's and using

$$\max_{t=1,\dots,T} \|\bar{x}_t - x_t\|_\infty \leq \max_{t=1,\dots,T} \operatorname{diam}_\infty(X(p_t)) \leq K,$$

we obtain

$$\frac{1}{T}\sum_{t=1}^{T} c_t^\top(\bar{x}_t - x_t) + \frac{1}{T}\sum_{t=1}^{T} c_{\text{true}}^\top(x_t - \bar{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}}.$$

Observe that for each summand $t \in [T]$ we have $c_t^\top(\bar{x}_t - x_t) \geq 0$ as $\bar{x}_t, x_t \in X(p_t)$ and $\bar{x}_t$ is the maximum over this set with respect to c_t. With a similar argument, we see that $c_{\text{true}}^\top(x_t - \bar{x}_t) \geq 0$ for all $t \in [T]$. Thus, we have

$$0 \leq \frac{1}{T}\sum_{t=1}^{T}(c_t - c_{\text{true}})^\top(\bar{x}_t - x_t) \leq 2K\sqrt{\frac{\ln n}{T}},$$

and similarly for the separate terms with analogue argumentation. This establishes the claim. □

Note that by using exponential updates of the form

$$w_{t+1}(i) \leftarrow w_t(i)e^{-\eta y_t(i)}$$

in Line 13 of the algorithm, we could attain essentially the same bound, cf. (Arora et al., 2012, Theorem 2.3). Secondly, we remark that our choice of the learning rate η requires the number of rounds T to be known beforehand; if this is not the case, we can use the standard doubling trick (see Cesa-Bianchi and Lugosi (2006)) or use an anytime variant of MWU.

From the above theorem, we can conclude that the average error over all observations (p_t, x_t) for $t = 1, \dots, T$ when choosing objective function c_t in iteration t of Algorithm 2 instead of c_{true} converges to 0 with an increasing number of observations T at a rate of roughly $\mathcal{O}(1/\sqrt{T})$:

Corollary 3.5. *Let $K \geq 0$ with $\operatorname{diam}_\infty X(p_t) \leq K$ for all $t = 1, \dots, T$. Then we have*

1. $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} c_t^\top(\bar{x}_t - x_t) = 0$ *and*
2. $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} c_{true}^\top(x_t - \bar{x}_t) = 0.$

In other words, both the average error incurred from replacing the actual objective function c_{true} by the estimation c_t as well as the average error in solution quality with respect to c_{true} tend to 0 as T grows.

Moreover, using Markov's inequality we also obtain the following quantitative bound on the deviation by more than a given $\varepsilon > 0$ from the average cost:

Corollary 3.6. *Let $\varepsilon > 0$. Then the fraction of observations x_t with*

$$c_{true}^\top(x_t - \bar{x}_t) \geq 2K\sqrt{\frac{\ln n}{T}} + \varepsilon$$

is at most

$$1 - \frac{\varepsilon}{2K\sqrt{\frac{\ln n}{T}} + \varepsilon}.$$

In particular, for any $0 < p < 1$ we have that after

$$T \geq \ln n \left(\frac{(1-p)2K}{p\varepsilon}\right)^2$$

observations the fraction of observations x_t with cost

$$c_{true}^\top(x_t - \bar{x}_t) \geq \frac{\varepsilon}{1-p} \geq 2K\sqrt{\frac{\ln n}{T}} + \varepsilon$$

is at most p.

Proof. Markov's inequality states

$$|\{x \in X \mid f(x) \geq a\}| \leq \frac{1}{a} \sum_{x \in X} |f(x)|$$

for a finite set X, a function $f\colon X \to \mathbb{R}$ and $a > 0$. With $X = [T]$, $f(t) = c_{\text{true}}^\top (x_t - \bar{x}_t)$ for $t \in [T]$ as well as $a = 2K\sqrt{(\ln n)/T} + \varepsilon$, we obtain the desired upper bound on the fraction of high deviations. The second part follows from solving

$$1 - \frac{\varepsilon}{2K\sqrt{\frac{\ln n}{T}} + \varepsilon} \leq p$$

for T and plugging in values. □

Remark 3.7. *It is straightforward to extend the result from Theorem 3.4 to a more general setup, namely the learning of an objective function which is linearly composed from a set of basis functions. To this end, we consider the problem*

$$\begin{aligned} \max \quad & c_{true}^\top f(x) \\ \text{s.t.} \quad & x \in X(p), \end{aligned}$$

where $c_{true} \in \mathbb{R}_+^n$ with $\|c_{true}\|_1 = 1$, $f\colon D \to \mathbb{R}^m$ on $D \subset \mathbb{R}^n$ compact and $X(p)$ parameterized in $p \in P$ as above. In order to apply Theorem 3.4 to this case, the ℓ_∞-diameter of the image of f additionally needs to be finite, which is naturally the case, for example, if f is Lipschitz continuous with respect to the maximum norm with Lipschitz constant L. Then we can change the cost function in Line 11 of a Algorithm 2 to

$$y_t = \frac{f(\bar{x}_t) - f(x_t)}{\|f(\bar{x}_t) - f(x_t)\|_\infty},$$

which yields a guarantee of

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{true})^\top (f(\bar{x}_t) - f(x_t)) \leq 2K\sqrt{\frac{\ln n}{T}},$$

with $K = L \cdot \max_{1,\dots,T} \operatorname{diam}_\infty(X(p_t))$.

We would like to point out that the requirement to observe optimal solutions x_t to learn the objective function c_{true} which produced them can be relaxed in all the above considerations. Assume that we observe $(1-\varepsilon)$-*optimal* solutions $\hat{x}_t \in X(p_t)$ instead, i.e. they satisfy $c_{\text{true}}^\top \hat{x}_t \geq (1-\varepsilon) c_{\text{true}}^\top x_t$ for all $t = 1, \dots, T$ and some $\varepsilon \geq 0$. In this case, the upper bound

$$\frac{1}{T} \sum_{t=1}^{T} (c_t - c_{\text{true}})^\top (\bar{x}_t - \hat{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}},$$

which is analoguous to what we derived in Theorem 3.4, still holds, as it does not depend on the optimality of the observed solutions. On the other hand, we have

$$\frac{1}{T} \sum_{t=1}^{T} (c_t - c_{\text{true}})^\top (\bar{x}_t - \hat{x}_t) \geq \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^\top (\hat{x}_t - \bar{x}_t) \geq \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^\top ((1-\varepsilon) x_t - \bar{x}_t)$$

due to the optimality of the $\bar{x}_t$'s with respect to the c_t's and the $(1-\varepsilon)$-optimality of the $\hat{x}_t$'s. Altogether, this yields

$$\frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^\top ((1-\varepsilon) x_t - \bar{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}}$$

and consequently

$$\frac{1}{T}\sum_{t=1}^{T} c_{\text{true}}^\top \bar{x}_t \geq \frac{1}{T}\sum_{t=1}^{T}(1-\varepsilon)c_{\text{true}}^\top x_t - 2K\sqrt{\frac{\ln n}{T}},$$

such that in the limit, our solutions x_t become $(1-\varepsilon)$-optimal on average. Note that a similar result can be obtained if we assume an additive error in the observed solutions $\hat{x}_t$ instead of a multiplicative one.

3.2 The Stable Case

While in most applications it is sufficient to be able to produce solutions via the surrogate objectives that are essentially equivalent to those for the true objective, we will show now that under slightly strengthened assumptions we can obtain significantly stronger guarantees for the convergence of the solutions: we will show that in the long run we learn to emulate the true optimal solutions provided that the problems have unique solutions as we will make precise now.

We say that the sequence of feasible regions $(X(p_t))_t$ is Δ-*stable for* c_{true} for some $\Delta > 0$ if for any $t \in [T]$, $c \in \mathbb{R}^n$ with $\|c\|_1 = 1$, $c \neq c_{\text{true}}$ and $\bar{x}_t \coloneqq \operatorname{argmin}\{c^\top x \mid x \in X(p_t)\}$ so that for $x_t \neq \bar{x}$ we have

$$c_{\text{true}}^\top(x_t - \bar{x}_t) \geq \Delta,$$

i.e. either the two optimal solutions coincide or they differ by at least Δ with respect to c_{true}. In particular, optimizing c_{true} over $X(p_t)$ leads to a unique optimal solution for all p_t with $t \in [T]$. While this condition – which is well known as the *sharpness* of a minimizer in convex optimization – sounds unnatural at first, it is, for example, trivially satisfied for the important case where $X(p_t)$ with $t \in [T]$ is a polytope with vertices in $\{0,1\}$ and c_{true} is a rational vector. In this case, write $c_{\text{true}} = d/\|d\|_1$ with $d \in \mathbb{Z}_+^n$ and observe that the minimum change in objective value between any two vertices x, y of the 0/1-polytope with $c_{\text{true}}^\top x \neq c_{\text{true}}^\top y$ is bounded by $|c_{\text{true}}^\top(x-y)| \geq 1/\|d\|_1$, so that Δ-stability with $\Delta \coloneqq 1/\|d\|_1$ holds in this case. The same argument works for more general polytopes via bounding the minimum non-zero change in objective function value via the encoding length.

We obtain the following simple corollary of Theorem 3.4.

Corollary 3.8. *Let* $K \geq 0$ *with* $\operatorname{diam}_\infty X(p_t) \leq K$ *for all* $t = 1, \ldots, T$, *let* $(X(p_t))_t$ *be* Δ*-stable for some* $\Delta > 0$, *and let* $N_T \coloneqq \{t \in [T] \mid \bar{x}_t \neq x_t\}$. *Then*

$$|N_T| \leq 2K\sqrt{\frac{T\ln n}{\Delta}}.$$

Proof. We start with the guarantee from the proof of Theorem 3.4:

$$0 \leq \frac{1}{T}\sum_{t=1}^{T} c_{\text{true}}^\top(x_t - \bar{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}}.$$

Now let N_T be as above so that

$$0 \leq \frac{1}{T}\sum_{t\in N_T} c_{\text{true}}^\top(x_t - \bar{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}}.$$

Observe that $\Delta \leq c_{\text{true}}^\top(x_t - \bar{x}_t)$ as x_t was optimal for c_{true} together with Δ-stability. We thus obtain

$$\frac{1}{T}|N_T|\Delta \leq 2K\sqrt{\frac{\ln n}{T}},$$

which is equivalent to

$$|N_T| \leq 2K\sqrt{\frac{T \ln n}{\Delta}}.$$

□

From the above corollary, we obtain in particular that in the Δ-stable case we have $\frac{1}{T}|N_T| \leq 2K\sqrt{(\ln n)/(T\Delta)}$, i.e. the average number of times that $\bar{x}_t$ deviates from x_t tends to 0 in the long run. We hasten to stress, however, that the convergence implied by this bound can potentially be slow as it is exponential in the actual encoding length of c_{true}; this is to be expected given the convergence rates of our algorithm and online-learning algorithms in general.

3.3 An Algorithm based on Online Gradient Descent

The algorithm based on MWU introduced in Section 3.1 has the limitation that it is only applicable for learning non-negative objectives. In addition, it cannot make use of any prior knowledge about the structure of c_{true} other than coming from the positive orthant. To lift these limitations, we will extend our approach using online gradient descent (OGD) which is an online-learning algorithm applicable to the following game over T rounds: in each round $t = 1, \ldots, T$, the player chooses a solution x_t from a convex, compact and non-empty feasible set $F \subset \mathbb{R}^n$. Then the adversary reveals to him a convex objective function $c_t \colon \mathbb{R}^n \to \mathbb{R}$, and the player incurs a cost of $c_t(x_t)$. OGD proceeds by choosing an arbitrary $x_1 \in F$ in the first round and updates this choice after observing c_t via

$$x_{t+1} = P(x_t - \eta_t \nabla c_t(x_t)),$$

where P is the projection onto the set F and η_t is the learning rate. With the abbreviations $D := \operatorname{diam}_2(F)$ and $G := \sup_{x \in F, t=1,\ldots,T} \|\nabla c_t(x)\|_2^2$, the regret of the player can then be bounded as follows.

Lemma 3.9 (Zinkevich (2003, Theorem 1))**.** *For $\eta_t = 1/\sqrt{t}$, $t = 1, \ldots, T$, we have*

$$\sum_{t=1}^{T} c_t(x_t) - \min_{x \in F} \sum_{t=1}^{T} c_t(x) \leq \frac{D^2\sqrt{T}}{2} + \left(\sqrt{T} - \frac{1}{2}\right) G^2.$$

Concerning the choice of learning rate, there are a couple of things to note. Firstly, the learning rate $\eta_t = 1/\sqrt{t}$ in round t does not depend on the total number of rounds T of the game. This means that the resulting version of OGD works without prior knowledge of T. It is even possible to improve slightly on the above result: by choosing the learning rate $\eta_t = D/(G\sqrt{t})$ in round t, the regret bound after T rounds becomes $(3/2)DG\sqrt{T}$, thus exhibiting smaller constant factors (see, for example, Hazan (2016, Theorem 3.1)). In the case of prior knowledge of T, it is possible to choose the constant learning rate $\eta_t = D/(G\sqrt{T})$ in each round t, which in our computational experiments leads to a smoother convergence especially in the first iterations and again marginally improves the regret bound; it is then possible to bound it by $DG\sqrt{T}$ (cf. the proof of Zinkevich (2003, Theorem 1)).

Again, we can reinterpret the underlying game of OGD in the context of learning objective functions. To this end, we swap the roles of the player and adversary, in the sense that the player now plays linear objective functions c_t from a set F and the adversary answers with the difference vectors $(\bar{x}_t - x_t)$ between their respective optimal solutions. This leads to the learning scheme described in Algorithm 3.

Obviously, this second algorithm is more general than the first one – we can now learn objective functions with arbitrary coefficients – but is also more computationally involved due to the projection step in Line 3. For suitably bounded sets F and $X(p_t)$, it yields the following performance guarantee which follows directly from Lemma 3.9 and the subsequent discussion on the choice of the learning rates.

Algorithm 3 Objective Function Learning via Online Gradient Descent

Input: observations (p_t, x_t) and learning rates $\eta_t = \frac{D}{G\sqrt{t}}$ with $t = 1, \dots, T$
Output: sequence of objectives $c_1, c_2, \dots, c_T$

choose $y_1 \in \mathbb{R}^n$ arbitrarily
for $t = 1, \dots, T$ **do**
  $c_t \leftarrow \operatorname{argmin} \{\|y_t - c\|_2 \mid c \in F\}$ {Project onto F}
  Observe parameters p_t
  $\bar{x}_t \leftarrow \operatorname{argmin} \{c_t^\top x \mid x \in X(p_t)\}$ {Solve Subproblem (2)}
  Observe expert solution x_t
  $y_t \leftarrow c_t - \eta_t(\bar{x}_t - x_t)$ {Perform gradient descent step}
end for
return $c_1, c_2, \dots, c_T$.

Theorem 3.10. *If* $\operatorname{diam}_2(F) \leq L$ *for some* $L \geq 0$ *and* $\operatorname{diam}_2(X_t) \leq K$, $t = 1, \dots, T$, *for some* $K \geq 0$, *then Algorithm 3 produces a series of objective functions* $(c_1, \dots, c_T)$ *with*

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{true})^\top (\bar{x}_t - x_t) \leq \frac{3LK}{2\sqrt{T}}.$$

Via this result, it is now not only possible to learn objective functions with arbitrary coefficients, and incorporate prior knowledge of its stucture, but we can also consider more general setups for learning objective functions as we demonstrate in the following.

Remark 3.11. *Using the above theorem along the lines of Remark 3.7, we can now learn a best-possible approximation of an arbitrary objective function f via a piecewise-defined function over a given triangulation of the feasible domain of f. As an example, we will consider learning a piecewise-linear approximation of a continuous objective function $f\colon [a, b] \to \mathbb{R}$ with n breakpoints. Let the breakpoints d_i with $i = 1, \dots n$ be such that $a = d_1 \leq \dots \leq d_n = b$. Then we can choose piecewise-defined basis functions of the form*

$$g_{ij}(x) = \begin{cases} x^j, & \text{if } x \in [d_i, d_{i+1}] \\ 0, & \text{otherwise} \end{cases}$$

with $i = 1, \dots n-1$ and $j = 0, \dots, j_{\max}$ for some desired maximal order $j_{\max}$. Our approximation of f will then be of the form $\sum_{i=1}^{n} \sum_{j=1}^{j_{\max}} c_{true,ij} g_{ij}(x)$. The set F from which c_{true} is assumed to originate can accordingly be chosen such that it models the boundary conditions for the continuity of the approximation via linear equations:

$$F := \{c \in R^{n \cdot (j_{\max}+1)} \mid \sum_{j=1}^{j_{\max}} c_{ij} g_{ij}(d_i) = \sum_{j=1}^{j_{\max}} c_{i+1,j} g_{i+1,j}(d_i), \quad i = 2, \dots, n-1\}.$$

This approach naturally generalizes to piecewise-defined functions in higher dimensions and higher orders of smoothness.

Using Theorem 3.10, it is also possible to learn linearly parameterized objective functions. To this end, we generalize $(OPT(p))_p$ by considering the family of problems $(OPT2(q, p))_{q,p}$ given by

$$\max \quad c_{\text{true}}(q)^\top x \tag{7a}$$
$$\text{s.t.} \quad x \in X(p), \tag{7b}$$

where c_{true} is now a linear function $c_{\text{true}}\colon Q \to \mathbb{R}^n$, $q \mapsto M_{\text{true}} q$ which depends on parameters $q \in Q \subset \mathbb{R}^m$ via multiplication with some matrix $M_{\text{true}} \in \mathbb{R}^{n \times m}$. The task is then to infer the

matrix M_{true} from the observed optimal solutions (again assuming that the model $X(p)$ of the feasible region is known).

First, observe that $c_{\text{true}}(q) = M_{\text{true}}q$ is equivalent to $c_{\text{true}}(q) = \sum_{i=1}^{m} q(i)c_{i,\text{true}}$ for some basic objective functions $c_{i,\text{true}} \in F$, $i = 1, \dots, m$. Defining $\text{vec}(M) := (c_{1,\text{true}}, \dots, c_{m,\text{true}})^\top \in F^m$ as the vector that arises by stacking the columns of M_{true}, and similarly defining $\text{vec}(q, x) := (q(1)x, \dots, q(m)x)^\top \in \mathbb{R}^{mn}$ for $x \in X(p_t)$, the objective function $c_{\text{true}}(q)^\top x$ of Problem (7) can also be written as $\text{vec}(M_{\text{true}})^\top \text{vec}(q, x)$. We now assume that in each round $t = 1, \dots, T$, in addition to the parameter realizations p_t determining the feasible region, we observe parameter realizations q_t determining the objective function according to the above construction. A direct application of Algorithm 3 then allows us to learn all the $c_{i,\text{true}}$'s simultaneously, yielding the following approximation guarantee:

Corollary 3.12. *Let the sets F and $X(p_t)$, $t \in [T]$, be as in Theorem 3.10. If there is an $N \geq 0$ with $\|q\|_2 \leq N$ for all $q \in Q$, Algorithm 3 produces a sequence of matrices $(\text{vec}(M_1), \dots, \text{vec}(M_T))$ with*

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (\text{vec}(M_t) - \text{vec}(M_{true}))^\top (\text{vec}(q_t, \bar{x}_t) - \text{vec}(q_t, x_t)) \leq \frac{3\sqrt{m}NLK}{2\sqrt{T}}.$$

Proof. This result directly follows from $\text{diam}_2(F^m) = \sqrt{m}\,\text{diam}_2(F) \leq \sqrt{m}L$ together with $\text{diam}_2(\bigtimes_{i=1}^{m} q_i X(p_t)) \leq N\,\text{diam}_2(X(p_t)) \leq NK$ for $t \in [T]$. □

A further extension of our approach is that of learning a dynamic objective function where there are no parameters known which determine how it changes from round to round. Naturally, this is only possible if the change in the true objective is suitably bounded. The following result for using online gradient descent to learn a dynamic strategy is the basis for an approximation guarantee:

Lemma 3.13 (Zinkevich (2003, Theorem 2))**.** *Let $\sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_2$ be the* path length *of a sequence $(x_1, \dots, x_T)$ with $x_t \in F$, $t = 1, \dots, T$, and let $\mathcal{X}(F, T, N)$ be the set of of all sequences of T vectors in F with path length at most $R \geq 0$. Under the same assumptions as for Lemma 3.9 and some fixed learning rate $\eta \in \mathbb{R}_+$, we have*

$$\sum_{t=1}^{T} c_t(x_t) - \min_{(\hat{x}_1, \dots, \hat{x}_T) \in \mathcal{X}(F,T,N)} \sum_{t=1}^{T} c_t(\hat{x}_t) \leq \frac{7R^2}{4\eta} + \frac{RD^2}{\eta} + \frac{T\eta G}{2}.$$

Choosing the fixed learning rate $\eta = D/(G\sqrt{T})$, we obtain an upper bound of order $\mathcal{O}(R\sqrt{T})$ in the above lemma, such that the average error vanishes if the path length grows slower asymptotically than $\sqrt{T}$. This directly translates into a guarantee for the regret when learning an dynamic objective function whose path length in F is bounded by some constant $R \geq 0$.

3.4 A Heuristic for Linear Programs

As laid out in Remark 3.1, we can easily learn objectives of linear programs by using LP duality. By solving Problem (3) for polyhedral feasible regions $X(p_t) = \{x \in \mathbb{R}^n \mid A_t x \leq b_t\}$ for $t = 1, \dots, T$ as well as a polyhedral region $F = \{c \in \mathbb{R}^n \mid Bc \leq d\}$, for the possible choices of c, we can determine an objective function that minimizes the total duality gap. In cases where the observations (p_t, x_t) are not given all at once but are revealed sequentially, it might be a useful heuristic to solve Problem (3) only for the data that has been revealed so far, and to use the resulting vector c as the objective function in the next round. Defining the set $Y(p_t, c) := \{y_t \in \mathbb{R}^n_+ \mid A_t^\top y_t = c\}$ for the parameters p_t in round t and $c \in F$ as an abbreviation, this yields Algorithm 4.

This method is a kind of follow-the-leader scheme (see Kalai and Vempala (2005); Hannan (1957)), where the objective is chosen such that it minimizes the total duality gap over all previous rounds (picking an arbitrary objective in the first round). For adversarially chosen p_t's, the average regret does not necessarily converge to 0, as the following counterexample shows.

Algorithm 4 Heuristic Objective Function Learning via Sequential LP Solving

Input: Observations (p_t, x_t) with $t = 1, \dots, T$
Output: Sequence of objectives $(c_1, c_2, \dots, c_T)$

for $t = 1, \dots, T$ **do**
  $c_t \leftarrow \operatorname{argmin} \{\sum_{\tau=1}^{t-1} (b_\tau^\top y_\tau - c^\top x_\tau) \mid y_\tau \in Y(p_\tau, c)\ (\forall \tau \in [t-1]),\ c \in F\}$
    {Solve Problem (3)}
  Observe parameters p_t
  $\bar{x}_t \leftarrow \operatorname{argmin} \{c_t^\top x \mid x \in X(p_t)\}$ {Solve Subproblem (2)}
  Observe expert solution x_t
end for
return $(c_1, c_2, \dots, c_T)$.

Example 3.14. *Consider the following series of feasible sets $X(p_t)$ over rounds $t = 1, \dots, T$ for some time horizon $T > 0$:*

$$X(p_t) := \{(x_1, x_2)^\top \in \mathbb{R}^2_+ \mid x_2 = 1 - \frac{1}{a_t}\},$$

where $a_t = \sum_{\tau=1}^{t} (\frac{1}{2})^{\tau-1}$. They represent line segements going from $(0,1)^\top$ to $(a_t, 0)^\top$ in round t, approaching the line segment from $(0,1)^\top$ to $(2,0)^\top$ with $T \to \infty$. The true objective to learn shall be $c_{true} = (0,1)^\top$ and $F = \Delta_2$. In this setting, the optimal solution of the adversary will always be $(0,1)^\top$. Assuming that the player chooses objective $c_1 = (1,0)^\top$ in the first round, his solution in that round will be $\bar{x}_t = (1,0)^\top$, while the adversary will play $x_t = (0,1)^\top$. In the next round, the player will thus have to play an objective whose angle to $(1,0)$ is at least as high as that of $(1,1)^\top$ to $(1,0)^\top$, i.e. a $c_2 = (\gamma, 1-\gamma)^\top$ with $\frac{1}{2} \leq \gamma \leq 1$. If, in general, the player chooses $c_t = (1/(1+a_{t-1}), a_{t-1}/(1+a_{t-1}))^\top$ in any round $t > 1$, he will always make $(0,1)^\top$ an optimal solution for all previously observed feasible sets $X(p_t)$, but will always pick $(a_t, 0)^\top$ as his optimal solution in the current round. He thus yields a regret of

$$\begin{aligned}
\sum_{t=1}^{T} c_t^\top (\bar{x}_t - x_t) - \sum_{t=1}^{T} c_{true}^\top (\bar{x}_t - x_t) &= \sum_{t=1}^{T} \begin{pmatrix} \frac{1}{1+a_{t-1}} \\ \frac{a_{t-1}}{1+a_{t-1}} \end{pmatrix}^\top \begin{pmatrix} a_t \\ -1 \end{pmatrix} - \sum_{t=1}^{T} \begin{pmatrix} 0 \\ 1 \end{pmatrix}^\top \begin{pmatrix} a_t \\ -1 \end{pmatrix} \\
&= \sum_{t=1}^{T} \frac{a_t - a_{t-1}}{1 + a_{t-1}} - \sum_{t=1}^{T} (-1) \\
&= \sum_{t=1}^{T} \frac{1}{2(1 + a_{t-1})} + T \\
&\geq \sum_{t=1}^{T} \frac{1}{6} + T \\
&= \frac{7T}{6},
\end{aligned}$$

such that neither of objective error, solution error or total error converges to zero on average with $T \to \infty$.

In addition to the possibly adversarial choice of p_t, usual regret bounds for follow-the-leader-style algorithms are not applicable here as the loss function in each round is conditional on the objective of the player. Note also that there is no element of randomness or regularization in the strategy of the player, which is usually required for convergence. Nevertheless, Algorithm 4 is an intuitive approach to take in the linear case as it terminates with an objective c_T that is an optimal fit for the training data. In our computational experiments, it performed well for stochastically sampled parameter realizations p_t. As with OGD, the set F, from which the true objective and the objectives of the player originate, can readily be chosen as an arbitrary set instead of a polyhedron as long as an oracle is available that can solve Problem (3) with Constraint (3d) suitably replaced.

3.5 Remarks on Learning Constraints

A natural question that arises is if the same methodology we have used to learn the objective of an optimization problem can be used to learn constraints as well. We will only briefly address this case here to indicate where some obstacles lie. We consider the family of optimization problems $(\text{OPT3}(p))_p$, $p \in P \subseteq \mathbb{R}^k$, given by

$$\begin{aligned} \max \quad & c(p)^\top x \\ \text{s.t.} \quad & Ax \leq b_{\text{true}} \\ & x \geq 0, \end{aligned}$$

where the objective function $c(p) \in \mathbb{R}^n$ depends on the parameters p, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix and $b_{\text{true}} \in \mathbb{R}^m$ is the right-hand side. Again, we assume that the learner observes pairs of parameter realizations and corresponding optimal solutions (p_t, x_t) in each round $t = 1, \ldots, T$. Furthermore, we assume that the objective functions $c(p_t)$ are known to both the learner and the expert. The same can be assumed for A without loss of generality by standard arguments. The right-hand side b_{true} is only known to the expert and to be learned from the observations.

The most natural approach for solving this learning problem is to apply Algorithm 2 to the dual of $\text{OPT2}(p_t)$,

$$\begin{aligned} \min \quad & b_{\text{true}}^\top y \\ \text{s.t.} \quad & A^\top y \geq c(p_t) \\ & y \geq 0, \end{aligned}$$

where y are the dual variables for the linear constraints. In the dual problem, b_{true} is the unknown objective function ($b_{\text{true}} \geq 0$ without loss of generality), while the constraints to be optimized over in each round are known – the same setting as before. It is important to note though that the learner has to observe the *dual* optimal solutions y_t and the guarantee will be that the *dual regret* is tending to 0. In addition, it remains open whether this scheme can directly be extended to also have the primal regret converge to 0; we suspect the answer to be in the negative in general.

4 Applications

We will now present several example applications of our framework for learning objective functions from observed decisions. These are the learning of customer preferences from observed purchases, the learning of travel times in a road network and the learning of optimal delivery routes. In each case, we will study differents assumptions for the nature of the objective function to learn in order to demonstrate the flexibility of our approach.

Our computational experiments have been conducted on a server comprising Intel Xeon E5-2690 3.00 GHz computers with 25 MB cache and 128 GB RAM. We have implemented our framework using the Python-API of *Gurobi 8.0.1* (see Gurobi Optimization, Inc. (2018)).

4.1 Learning Customer Preferences

We consider a market where different goods can be bought by its customers. The prices for the goods can vary over time, and we assume that the goods are chosen by the customers to maximize their utility given their respective budget constraints. Each sample (p_t, x_t) corresponds to a customer $t \in [T]$, where $p_t = (p_{t0}, p_{t1}, \ldots, p_{tn})$ contains his budget $p_{t0} \geq 0$ and the

current prices $p_{ti} \geq 0$ for each good $i \in [n]$. Customer t is then assumed to solve the following optimization problem $\mathrm{OPT}(p_t)$:

$$\begin{aligned} \max \quad & \sum_{i \in [n]} u_i x_i \\ \text{s.t.} \quad & \sum_{i \in [n]} p_{ti} x_i \quad \leq \quad p_{t0} \\ & x \quad \in \quad \{0,1\}^n, \end{aligned}$$

where the aggregate utilities $u_i \geq 0$ of good i to the customers are unknown. Learning these utilities can for example help stores to take suitable assortment choices.

We consider two different setups: in the first setup, we assume that the goods are divisible, which means that the condition $x \in \{0,1\}^n$ is relaxed to $x \in [0,1]^n$; this is the *linear knapsack problem*. In the second setup, the goods are indivisible, so that we solve the problem with the original constraint $x \in \{0,1\}^n$ as an integer program; this is the *integer knapsack problem*.

To simulate the first setup, we generated 50 random instances, in each instance considering $T = 500$ observations for $n = 50$ goods. The customers' unknown utilities for the different goods are drawn as integer numbers from the interval $[1, 1000]$ according to a uniform distribution and then normalized in the 1-norm. The prices for sample t are chosen to be $p_{ti} = u_i + 100 + r_{ti}$, $i \in [n]$, where r_{ti} is an integer uniformly drawn from the interval $[-10, 10]$. Finally, the right-hand side p_{t0} is again an integer drawn uniformly from the interval $[1, \sum_{i \in [n]} p_{ti} - 1]$. Choosing utilities and weights in a strongly correlated fashion as above typically leads to harder (integer) knapsack problems, see Pisinger (2005) for more details. Note, however, that the focus here is not the hardness but rather the non-triviality of the instances.

In the following, we study learning the utilities for the linear knapsack problem using three different algorithms: Algorithm 2 based on MWU, Algorithm 3 based on OGD with $F = \Delta_n$ and the constant learning rate $\eta_t = D/(G\sqrt{T})$, as well as Algorithm 4 based on sequential LP solving, where the latter is of course only applicable in the linear case.

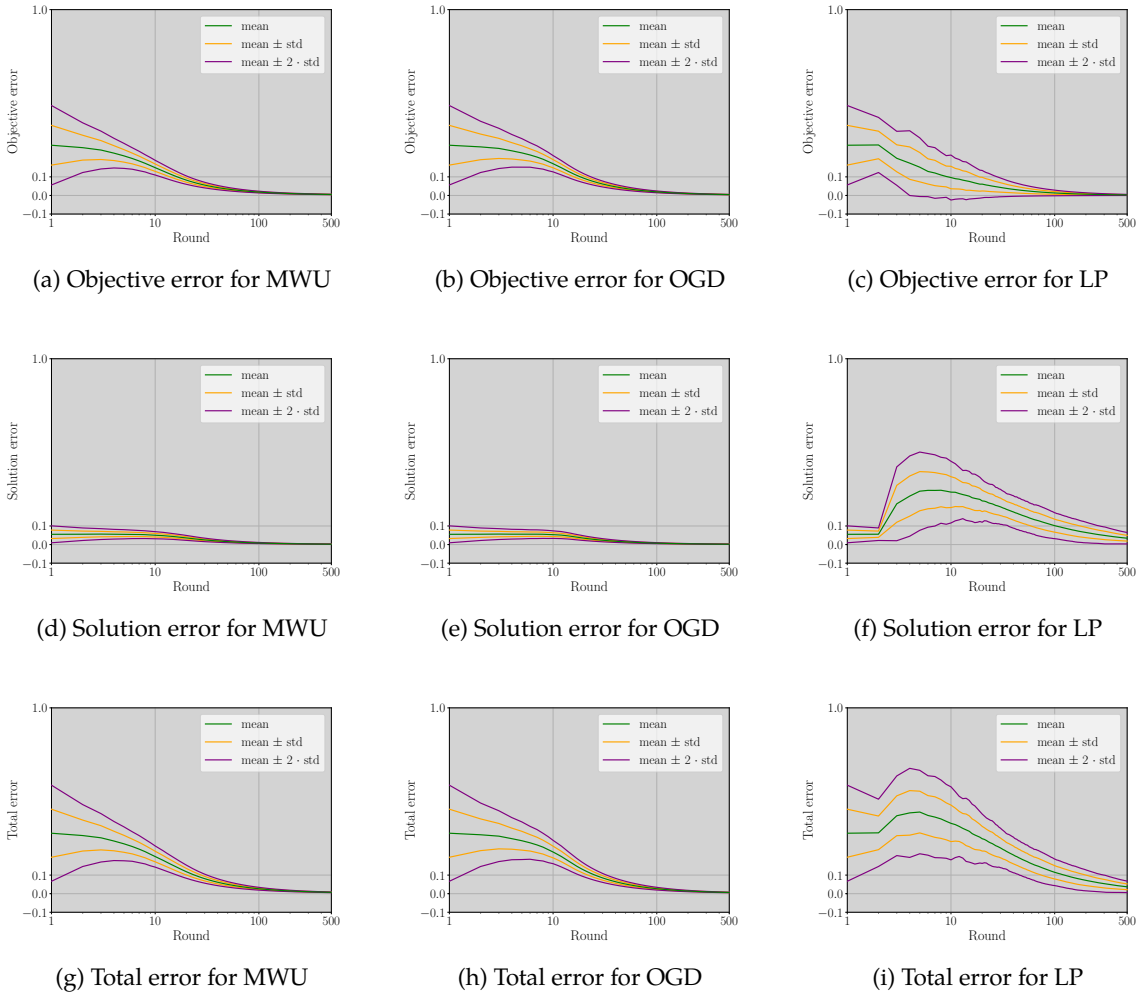
Algorithm	MWU			OGD			LP		
#Rounds	5	50	500	5	50	500	5	50	500
Mean avg. objective error	0.21	0.03	0.01	0.23	0.04	<0.01	0.15	0.03	<0.01
Std. of avg. objective error	0.03	<0.01	<0.01	0.04	<0.01	<0.01	0.08	0.01	<0.01
Mean avg. solution error	0.06	0.02	<0.01	0.06	0.01	<0.01	0.29	0.16	0.03
Std. of avg. solution error	0.01	<0.01	<0.01	0.01	<0.01	<0.01	0.11	0.04	0.02
Mean avg. total error	0.27	0.05	0.01	0.28	0.05	0.01	0.44	0.18	0.04
Std. of avg. total error	0.05	0.01	<0.01	0.05	0.01	<0.01	0.11	0.04	0.02

Table 1: Statistics for the average errors for the linear knapsack problem with $n = 50$ items and $T = 500$ observations for each algorithm, rounded to two decimals, with the arithmetic mean taken over 50 runs on random instances

In Table 1, we show statistics on the computational results for the linear knapsack problem. It shows the arithmetic means and standard deviations of the average errors after 5, 50 and 500 iterations for each of the algorithms, with the arithmetic mean taken over all 50 instances. Recall that in Section 3, we defined the following error types: the objective error for each round $t \in [T]$ is defined by $c_t^\top(\bar{x}_t - x_t)$ and describes the deviation between the solution $\bar{x}_t$ found by the oracle in that round and the solution x_t observed from the expert as evaluated with our guess for the objective function c_t. Accordingly, the solution error in each round t is defined as $c_{\text{true}}^\top(\bar{x}_t - x_t)$ and evaluates the deviation between the two solutions measured in the true objective function. Together they yield the total error, given by $(c_t - c_{\text{true}})^\top(\bar{x}_t - x_t)$, which is

the total deviation between choosing c_t and c_{true} in Problem (1) in round t. We can see that the average errors converge to 0 rather quickly, such that it is possible to take practically optimal decision after 500 rounds in all cases – with MWU and OGD performing very similary to each other, which is explainable by the fact that the algorithms are basically the same, except for the difference in the projection step. Both performing notably better in early iterations than LP, which is, however, able to catch up, as after a couple of rounds the solution error is basically always zero.

Each of these error types is also shown in our plots over all rounds in Figure 1. For each



(a) Objective error for MWU (b) Objective error for OGD (c) Objective error for LP

(d) Solution error for MWU (e) Solution error for OGD (f) Solution error for LP

(g) Total error for MWU (h) Total error for OGD (i) Total error for LP

Figure 1: The arithmetic means and standard deviations of the different error types in each round (*red*) and averaged up to the current round (*blue*) for the linear knapsack problem with $n = 50$ items over $T = 500$ rounds for each of the three algorithms, with the arithmetic mean taken over 500 runs, on a doubly symmetric-logarithmic plot

algorithm, they depict the arithmetic mean of the average objective, solution and total error over the 50 instances, together with the first and second standard deviation. As can be seen in general, after few iterations most error values reside close to zero, and the standard deviations lower quickly with the number of rounds played. The pictures for MWU and OGD are almost indistinguishable, while the mean average errors for LP are on a somewhat larger scale due to the tendentially higher errors in the very first rounds.

We also conducted an experiment for the integer knapsack problem, using $n = 1000$ and $T = 1000$. This time, we considered a single instance run with MWU as well as two different

versions of OGD: one with a fixed and one with a dynamic learning rate. From Table 1, which shows the average errors after 10, 100 and 1000 rounds, we see that the behaviour of the algorithms is virtually the same as for the linear knapsack, which means that the learning task does not become significantly harder because of the integrality requirement. Figure 2 shows

Algorithm	MWU			OGD (fixed)			OGD (dynamic)		
Number of rounds	10	100	1000	10	100	1000	10	100	1000
Average objective error	0.15	0.02	<0.01	0.20	0.02	<0.01	0.05	0.01	<0.01
Average solution error	0.05	0.01	<0.01	0.06	0.01	<0.01	0.03	0.01	<0.01
Average total error	0.20	0.03	<0.01	0.26	0.03	<0.01	0.08	0.02	<0.01

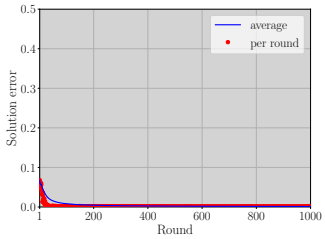
Table 2: Average errors for the integer knapsack problem with $n = 1000$ items

the corresponding error plots over all rounds, depicting both the error in a given round t and the average total error up to round t. Again, the average errors fall quickly, and only few of the errors in indidual rounds, mainly in the first rounds, deviate beyond the average.

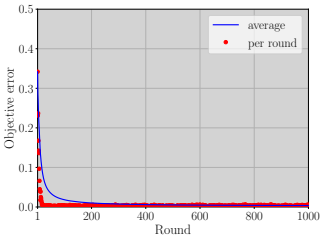
Next, we study the change in the learned objective function over time at the example of the integer knapsack problem. In Figure 3, we compare the convergence behaviour of the learned objective in the 1-norm for the three algorithms. It is visible at first sight that empirically they do not converge to the true objective function. In the case of MWU, the distance to the true objective function converges to about 0.1. For OGD with fixed learning rate $\eta = D/G\sqrt{T}$, the behaviour is very similar. When we choose the dynamic learning rate $\eta_t = D/G\sqrt{t}$, with a higher scale of the updates at the beginning, the distance convergence is slower by a factor two, which is in line with our treatment of the choice of learning rates in Section 3.3. The observed convergence to an alternative objective function is explainable by our discussion in Example 3.2, as there might be many different objective functions explaining the same observed solutions. From the previous table and the corresponding graphics, we have already seen that the alternative objectives we find perform about equally well than the original one.

A related question of interest is what the actual speed of convergence of the three algorithms is in comparison to their proven aymptotic behaviour. Figure 4 shows the same algorithms for the integer knapsack, depicting the average total error up to a given round versus the asymptotic upper bound represented by the function $f(T) = 1/\sqrt{T}$ on a log-log-scale. What we observe is that – over the total observation horizon – the order of convergence is basically same as that of f, where, again, all algorithms perform about equally well.

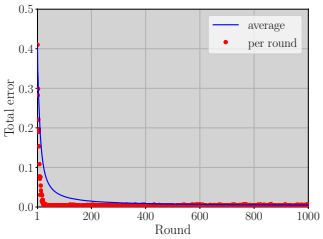
Finally, we investigate the out-of-sample performances of several policies for continuing the optimization if we receive no further feedback after a given point in time. To this end, we show in Figure 5 the solution error for the objectives produced by MWU for the case that there are no further updates after 100 rounds. In Figure 5a, we use c_{100} for the remaining 900 rounds, in Figure 5b, we use $\frac{1}{100}\sum_{t=1}^{100} c_t$, in Figure 5c, it is the c_t which produced the lowest error in the corresponding round t, over all rounds $t = 0, \ldots, 100$, and in Figure 5d we revert to $\mathbb{1}^n$ as the objective as a cross-check. We find that c_{100} consistently produces solutions with a cost similar to that of the true objective and thus generalizes very well to the unseen data in this instance. The same holds for the best c_t, which in this case was highly non-unique, as there are many rounds where the error is 0. Thus, we averaged over all c_t with error 0 and chose the resulting objective. In contrast, averaging over all c_t from the first 100 rounds performs much worse, and reverting to $\mathbb{1}^n$ expectably deterioriates the result, as all previous training is discarded. Altogether, we conclude that our approach leads to objective functions which provide a consistent explanation for the observed decisions, and in settings with i.i.d. sampled parameters p_t empirically even for those observations which the algorithm has not seen before.



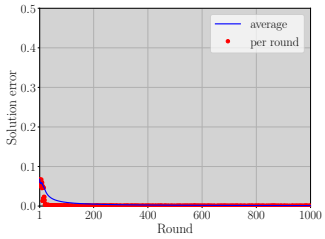
(a) Solution error for MWU



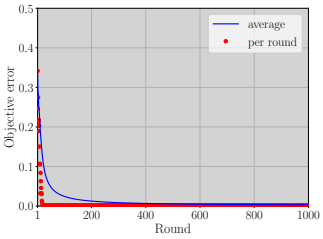
(b) Objective error for MWU



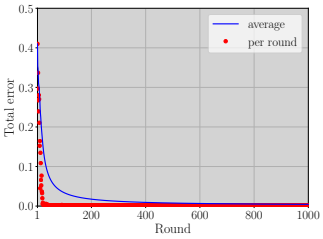
(c) Total error for MWU



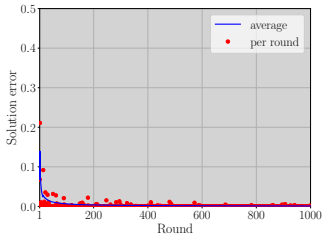
(d) Solution error for OGD with fixed learning rate



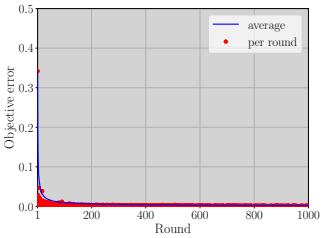
(e) Objective error for OGD with fixed learning rate



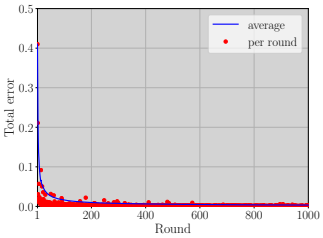
(f) Total error for OGD with fixed learning rate



(g) Solution error for OGD with dynamic learning rate



(h) Objective error for OGD with dynamic learning rate



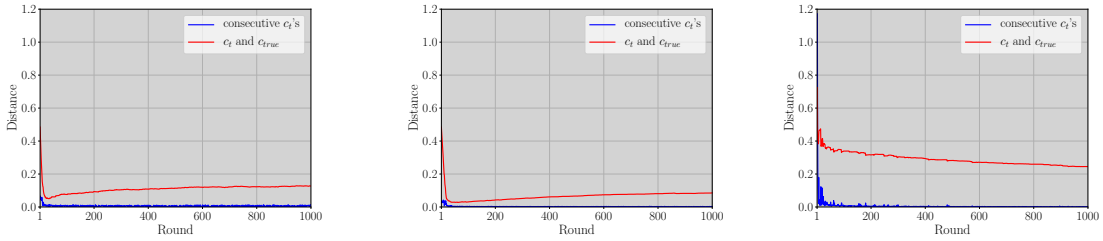
(i) Total error for OGD with dynamic learning rate

Figure 2: The values of the different error types for each round (*red*) and averaged up to the current round (*blue*) for the integer knapsack problem with $n = 1000$ items over $T = 1000$ rounds for MWU and two variants of OGD

4.2 Learning Travel Times

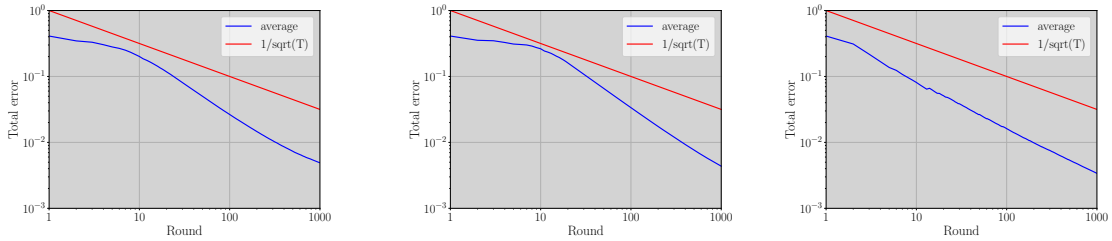
In our second computational experiment, we consider a street network where the travel times on a segment may vary over the day. Each driver in the network is assumed to choose a route that leads him from his origin to his destination in the shortest time possible. In other words, each driver solves a shortest-path problem on the same directed graph $G = (V, A)$. An observation $p_t = (p_t^1, p_t^2)$ for $t \in [T]$ represents a driver in the network who departs at a given time step in the observation period which we also denote by t, going from the starting point $p_t^1 \in V$ to the end point $p_t^2 \in V$. The entries of x_t indicate the path taken by driver t, which he obtains by solving the following optimization problem OPT(p_t):

$$\begin{aligned} \min \quad & \sum_{a \in A} c_{\text{true},ta} x_{ta} \\ \text{s.t.} \quad & \sum_{a \in \delta^+(v)} x_{ta} - \sum_{a \in \delta^-(v)} x_{ta} = \begin{cases} 1, & \text{if } v = p_t^1 \\ -1, & \text{if } v = p_t^2 \\ 0, & \text{otherwise} \end{cases} \qquad (\forall v \in V) \\ & x_{ta} \in \{0,1\}^{|A|}. \end{aligned}$$



(a) MWU (b) OGD with fixed learning rate (c) OGD with dynamic learning rate

Figure 3: Distance in the 1-norm between two consecutive objective objective functions (*blue*) and between the objective function in the current round and the true objective function (*red*) for the integer knapsack problem



(a) MWU (b) OGD with fixed learning rate (c) OGD with dynamic learning rate

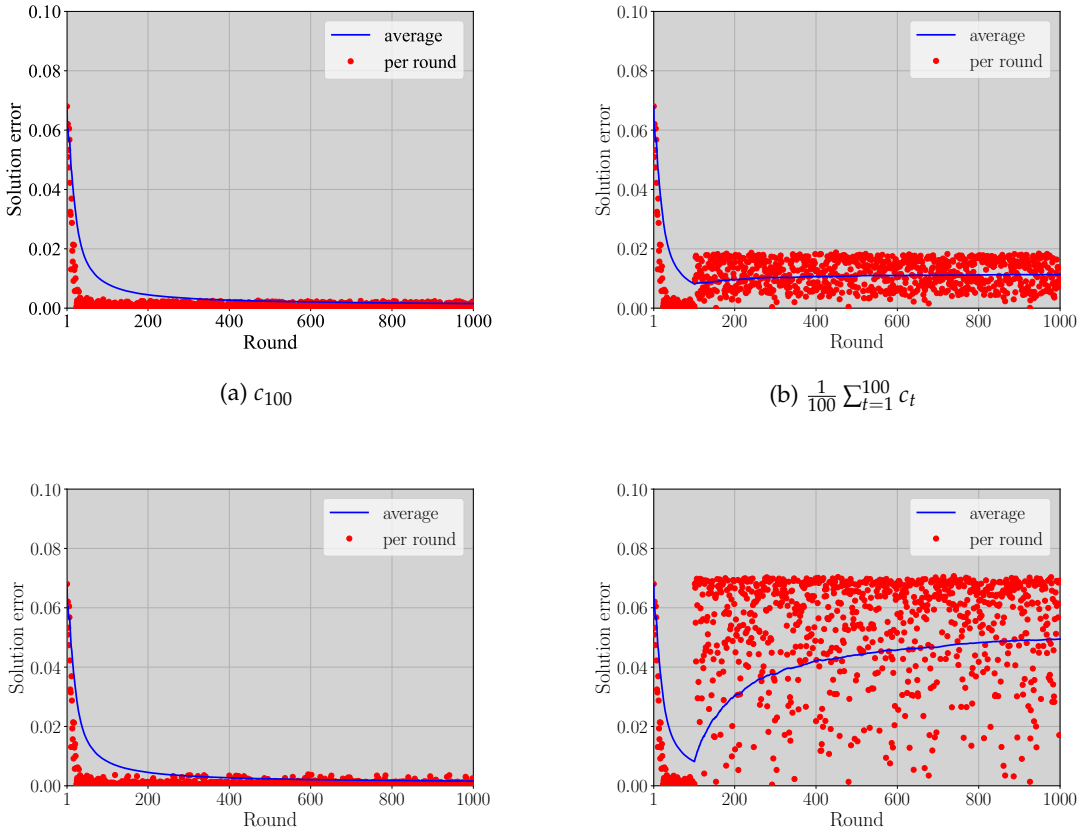
Figure 4: Doubly logarithmic plot of the average total error (*blue*) for the integer knapsack problem against the function $f(T) = 1/\sqrt{T}$ (*red*)

Observing the paths of each of the drivers, we want to learn the values $c_{\text{true},ta}$ corresponding to the travel times to traverse arc $a \in A$ at time step t. The major difference to our previous experiment is that here the objective function will be allowed to change over time, representing, for example, slowdowns due to traffic congestions.

We created instances of the problem based on a real-world street network, namely an aggregated version of the city map of Chicago. It is available as instance *ChicacoSketch* in Ben Stabler's library of transportation networks (see Stabler (2018)), and has 933 nodes and 2950 arcs (of which we ignore the 387 nodes representing "zones" as well as their incident arcs). Each arc a has a certain free-flow time $c_{\text{free},a}$ which we assume to be the unknown uncongested travel time. For each driver t, we chose a random pair of an origin and a destination node. Furthermore, we consider 5 hours of real time and one driver per 5 minutes for creating the observations. Hence, our time horizon is $T = 60$, which is also the number of drivers.

We consider three different settings. In a first step, we try to retain the free-flow times from the observed paths, i.e. we assume the travel times to be constant: $c_{\text{true},ta} := c_{\text{free},a}$ for all $a \in A$ and $t = 1, \ldots, T$.

As a second test, we integrate temporary increases in the travel time induced by congestions on the most-frequently used arcs. These increases were modelled in the following way: we computed all shortest paths in the network between any two nodes and chose the top 5% of arcs with the highest number of shortest paths traversing them as bottlenecks. For each of



(a) c_{100}

(b) $\frac{1}{100}\sum_{t=1}^{100} c_t$

(c) Best c_t out of $t = 1, \ldots, 100$

(d) Revert to $\mathbb{1}^n$

Figure 5: Comparison of the solution error for four policies to continue the optimization if the algorithm (here: MWU) receives no further update after $t = 100$ rounds

these bottlenecks a, we chose the travel time at time step t to be

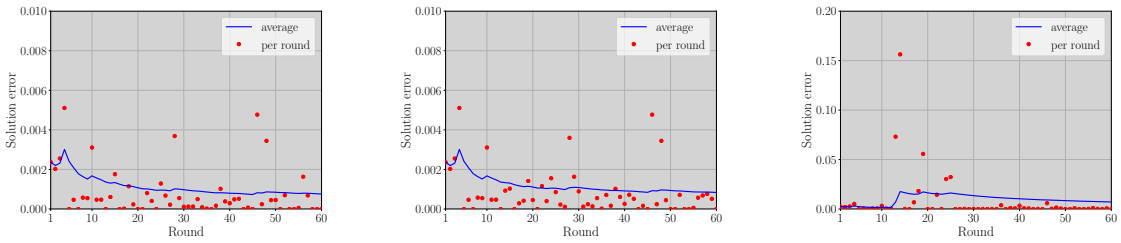
$$c_{\text{true},ta} := \begin{cases} c_{\text{free},a}, & \text{if } t \in \{1, \ldots, 11\} \\ (1 + \frac{t-12}{6})c_{\text{free},a}, & \text{if } t \in \{12, \ldots, 18\} \\ 2c_{\text{free},a}, & \text{if } t \in \{19, \ldots, 29\} \\ (1 + \frac{36-t}{6})c_{\text{free},a}, & \text{if } t \in \{30, \ldots, 36\} \\ c_{\text{free},a}, & \text{if } t \in \{37, \ldots, 60\}. \end{cases}$$

This means that at one hour of real time, the congestions on all bottleneck arcs a start to build up and reach the maximal congestion within 30 minutes, staying on maximal congestion for one hour and then ebbing away within 30 minutes.

Finally, we consider the case of abrupt changes in travel time as they might arise from roads which are suddenly blocked for some reason. To this end, we altered the travel times of the same arcs as before, but chose the travel time at time step t as

$$c_{\text{true},ta} := \begin{cases} 1000c_{\text{free},a}, & \text{if } t \in \{12, \ldots, 36\} \\ c_{\text{free},a}, & \text{otherwise.} \end{cases}$$

We then used MWU to learn the dynamically changing travel times. Figure 6 depicts the solution error for the three cases. From Figure 6a, we can see that we achieve a very good convergence of the error already with as few as 60 observations. For a gradually building and receding congestion, we see in Figure 6b that the performance of our algorithm nearly does



(a) No congestion (b) Gradual congestion (c) Abrupt congestion

Figure 6: Solution error for each round (*red*) and averaged up to the current round (*blue*) for the shortest-path problem with $T = 60$ drivers under the different congestion times for MWU

not deterioriate at all, which means the learned objective quickly adapts to the slowly changing travel times. In the case of abrupt congestions, we see the error spiking sharply at the point where the congestions begin, but quickly declining afterwards. After 60 rounds, the average solution error is still about 8 times higher than in the unpertubed case, which means it takes some time to recover. This behaviour is expected, as our theoretical results predict the washing out of any error d in the average regret at a rate of $\mathcal{O}(d/\sqrt{T})$, cf. the discussion of Lemma 3.13. Nevertheless, the solutions we obtain from round 26 on have a loss which is significantly below the average regret. This shows that we could quickly adapt even after a major disruption in the learning target.

In the following, we give a visualization of the solutions produced by the original as well as the learned objective functions. Figure 7a shows the graph corresponding to the Chicago street network, where we have only plotted the arcs which were actually used in a any of the shortest paths over 60 rounds, assuming the constant free-flow objective. The darker an arc is coloured, the higher is the number of actually taken paths which it is part of. We can clearly recognize several highways as well as shortcuts through the city center. In Figure 7b, we show the same figure, but with the paths learned by MWU without congestions. We see that the algorithm chooses a couple of routes which would not be chosen according to the true objective, expectably at the beginning, as it first needs to learn which ones are the good arcs (starting from the assumption that all arcs are of equal travel time). We clearly observe that the most-frequently used arcs are the same ones as with the true objective. The same holds for the cases of gradual and abrupt congestions, shown in Figures 7c and 7d respectively, with the exception that the most-frequent arcs are not chosen as often, as they take longer to traverse or are even blocked for a considerable amount of time. Instead, a few arcs belonging to detours become more interesting. To summarize, this experiment shows that we can learn to take efficient decisions already with few observations at hand and that we can quite easily cope with small continuing changes or big but seldom changes in the target objective.

4.3 Learning Optimal Delivery Routes

Finally, we demonstrate the potential of OGD to learn objective coefficients with mixed signs. For this purpose, we consider a simple delivery problem where a company has certain customers which it serves from its depot. The delivery network is given by a complete undirected graph $G = (V \cup \{v_0\}, E)$ with a designated node v_0 representing the depot and the nodes in V representing the customers. A delivery route consists of a Hamilton tour comprising the depot as well as the subset of customers the company decides to serve. The company has to decide in each time step which customers to serve in which order. We assume that each edge $e \in E$ has costs c_{te} for traversing it, and that each customer $v \in V$ brings a revenue r_{tv} if the company decides to serve him, both unknown and varying over time steps $t \in [T]$. This un-

(a) True objective

(b) No congestion

(c) Gradual congestion

(d) Abrupt congestion

Figure 7: Actual paths taken in the network for the true objective and the objectives learned by MWU under the three different congestion types

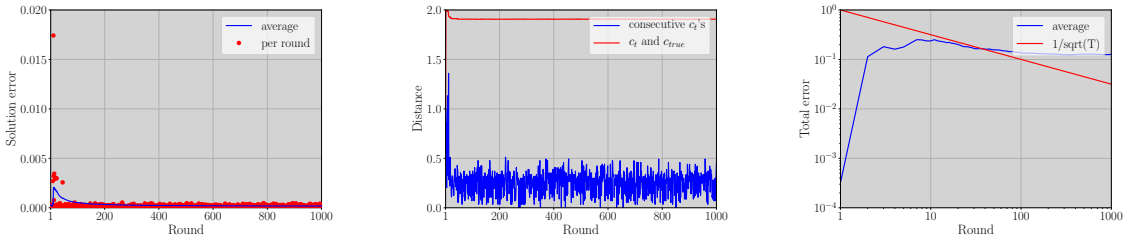
certainty might for example arise from different traffic scenarios affecting delivery costs as well as different customer demand scenarios. Altogether, the company wants to solve the following profitable tour problem OPT(p_t), a prize-collecting variant of the travelling salesman problem,

for each $t \in [T]$:

$$
\begin{aligned}
\max \quad & \sum_{v \in V} r_{tv} y_{tv} - \sum_{e \in E} c_{te} x_{te} \\
\text{s.t.} \quad & \sum_{e \in \delta(v)} x_{te} = 2 y_{tv} && (\forall v \in V \cup \{v_0\}) \\
& \sum_{\substack{e=(i,j) \in E: \\ i,j \in S}} x_{te} + y_l \leq 1 + \sum_{i \in S \setminus \{k\}} y_i && (\forall S \subset V \cup \{v_0\}, 2 \leq |S| \leq |V| - 1) \\
& && (\forall k \in S)(\forall l \notin S) \\
& y_{tv} \leq y_{tv_0} && (\forall v \in V) \\
& x_t \in \{0,1\}^{|E|} \\
& y_t \in \{0,1\}^{|V|+1},
\end{aligned}
$$

where the x-variables model the chosen edges and the y-variables model the chosen customers.

For our computational experiment, we use instance *berlin52-gen3-50* from Gorka Kobeaga's OPLib, see Kobeaga (2018), where we scale up the customer revenues by a factor of 4 to yield a non-trivial trade-off against the edge costs. Then in each time step $t \in [T]$, we draw the edge costs uniformly from an interval of $\pm 10\%$ and the customer revenues from an interval of $\pm 20\%$ around these basic values which we interpret as the expected costs and revenues respectively. Thus, we want to learn to distinguish the more efficient routes from the less efficient routes and the more profitable customers from the less profitable customers. We choose F to be the unit simplex and c_1 as the zero vector to initialize the OGD algorithm.



(a) Solution error (b) 1-norm objective distances (c) Convergence

Figure 8: Solution error, objective distances and convergence for the prize-collecting travelling salesman problem for OGD

Figure 8 shows the results of the experiment. In Figure 8a, we see that the solution error falls relatively quickly after few iterations as for the other two problems. However, now the average regret does not converge to zero, but rather to the variance in the true objective which is sampled anew in each round. This is explainable by the dynamic nature of the objective function to be learned, cf. Lemma 3.13 and the corresponding discussion. We obtain a kind of "robust" objective, which tries to explain the observations produced by an actually non-constant target objective.

5 Final Remarks

We saw that algorithms derived from online-learning methods are capable of learning objective functions from optimal (and close-to-optimal) observed decisions, given knowledge of the underlying feasible set in each case. We were able to prove that these algorithms achieve low errors, and we demonstrated that they quickly converge in computational experiments, learning objectives which explain the observed decisions very well, including when testing them

out-of-sample. They are applicable in more general situations than previous methods, which required convexity of the feasible region, and they are usable in situations where the observations arrive online as a data stream, allowing to learn objectives which change over time. The practical importance of this approach is evident when considering that much effort is made to come to procedures that automate model building from data. An important question in this respect is to what extent our framework can be extended to the learning of constraints (and objective functions simultaneously).

Acknowledgements

Research reported in this paper was partially supported by NSF CAREER award CMMI-1452463.

References

Abbasi-yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24*, pages 2312–2320. Curran Associates, Inc. Available at: `http://papers.nips.cc/paper/4417-improved-algorithms-for-linear-stochastic-bandits.pdf`.

Ahuja, R. K. and Orlin, J. B. (2000). A faster algorithm for the inverse spanning tree problem. *Journal of Algorithms*, 34(1):177–193.

Arora, S., Hazan, E., and Kale, S. (2012). The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8:121–164.

Aswani, A., Shen, Z.-J. M., and Siddiq, A. (2018). Inverse optimization with noisy data. *Operations Research*, 66(3):870–892.

Audibert, J.-Y., Bubeck, S., and Lugosi, G. (2013). Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45.

Bärmann, A., Pokutta, S., and Schneider, O. (2017). Emulating the expert: Inverse optimization through online learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 400–410. PMLR. Available at: `http://proceedings.mlr.press/v70/barmann17a.html`.

Bertsimas, D. and Kallus, N. (2016). Pricing from observational data. Technical report, Massachusetts Institute of Technology. available at:`https://arxiv.org/pdf/1605.02347`.

Burton, D. and Toint, P. L. (1992). On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53(1):45–61.

Burton, D. and Toint, P. L. (1994). On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63(1):1–22.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.

Chan, T. C. Y., Lee, T., and Terekhov, D. (2018). Inverse optimization: Closed-form solutions, geometry, and goodness of fit. *Management Science*. To appear.

Chen, X., Owen, Z., Pixton, C., and Simchi-Levi, D. (2015). A statistical learning approach to personalization in revenue management. Technical report, New York University. Available at: `https://papers.ssrn.com/sol3/papers2.cfm?abstract_id=2579462`.

D. Burton, W. R. Pulleyblank, P. L. T. (1997). *Network Optimization*, chapter The Inverse Shortest Paths Problem with Upper Bounds on Shortest Paths Costs, pages 156–171. Springer.

Dani, V., Hayes, T. P., and Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory (COLT)*. Available at: http://colt2008.cs.helsinki.fi/papers/80-Dani.pdf.

Daumé, H., Khuller, S., Purohit, M., and Sanders, G. (2005). On correcting inputs: Inverse optimization for online structured prediction. In *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. Available at: http://drops.dagstuhl.de/opus/volltexte/2015/5637/pdf/27.pdf.

den Hertog, D. and Postek, K. (2016). Bridging the gap between predictive and prescriptive analytics – new optimization methodology needed. Technical report, Tilburg University, Netherlands. Available at: http://www.optimization-online.org/DB_HTML/2016/12/5779.html.

Esfahani, P. M., Shafieezadeh-Abadeh, S., Hanasusanto, G. A., and Kuhn, D. (2018). Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167(1):191–234.

Freund, Y. and Schapire, R. E. (1997). Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103.

Gurobi Optimization, Inc. (2018). Gurobi optimizer reference manual. http://www.gurobi.com.

Hannan, J. (1957). Approximation to bayes risk in repeated play. In *Contributions to the Theory of Games*, volume 3, pages 97–139. Princeton University Press.

Hazan, E. (2016). *Introduction to Online Convex Optimization*. Now Publishers, Inc.

Iyengar, G. and Kang, W. (2005). Inverse conic programming with applications. *Operations Research Letters*, 33(3):319–330.

Jadbabaie, A., Rakhlin, A., Shahrampour, S., and Sridharan, K. (2015). Online optimization: Competing with dynamic comparators. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 398–406. PMLR. Available at: http://proceedings.mlr.press/v38/jadbabaie15.html.

Kalai, A. and Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71:291–307.

Kallus, N. and Udell, M. (2015). Learning preferences from assortment choices in a heterogeneous population. Technical report, Massachusetts Institute of Technology. available at: https://pdfs.semanticscholar.org/b29d/026b6776e94a00d1ea15f83518ebbbd14d85.pdf.

Kallus, N. and Udell, M. (2016). Dynamic assortment personalization in high dimensions. Technical report, Cornell University. Available at: https://arxiv.org/pdf/1610.05604.

Keshavarz, A., Wang, Y., and Boyd, S. (2011). Imputing a convex objective function. In *Proceedings of the 2011 IEEE International Symposium on Intelligent Control (ISIC)*, pages 613–619. Available at: https://ieeexplore.ieee.org/document/6045410.

Kobeaga, G. (2018). OPLib. Available at: https://github.com/bcamath-ds/OPLib.

Konstantakopoulos, I. C., Ratliff, L. J., Jin, M., Spanos, C., and Sastry, S. S. (2016). Smart building energy efficiency via social game: A robust utility learning framework for closing-the-loop. In *2016 1st International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE) in partnership with Global City Teams Challenge (GCTC) (SCOPE - GCTC)*, pages 1–6. Available at: https://ieeexplore.ieee.org/document/7515054.

Li, J. Y.-M. (2016). Inverse optimization of convex risk function. Technical report, University of Ottawa, Canada. available at: https://arxiv.org/abs/1607.07099.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, 108:212–261.

Lodi, A. (2016). Big data & mixed-integer (nonlinear) programming. Presentation, available at: https://atienergyworkshop.files.wordpress.com/2015/11/andrealodi.pdf.

Molloy, T. L., Tsai, D., Ford, J. J., and Perez, T. (2016). Discrete-time inverse optimal control with partial-state information: A soft-optimality approach with constrained state estimation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1926–1932. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7798546.

Nielsen, T. D. and Jensen, F. V. (2004). Learning a decision makers's utility function from (possibly) inconsistent behavior. *Artificial Intelligence*, 160:53–78.

Panchea, A. M. and Ramdani, N. (2015). Towards solving inverse optimal control in a bounded-error framework. In *2015 American Control Conference (ACC)*, pages 4910–4915. Available at: https://ieeexplore.ieee.org/document/7172103.

Papadopoulos, A. V., Bascetta, L., and Ferretti, G. (2016). Generation of human walking paths. *Autonomous Robots*, 40(1):55–75.

Pisinger, D. (2005). Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284.

Plotkin, S. A., Shmoys, D. B., and Éva Tardos (1995). Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301.

Qiang, S. and Bayati, M. (2016). Dynamic pricing with demand covariates. Technical report, Stanford University. available at: https://papers.ssrn.com/sol3/papers2.cfm?abstract_id=2765257.

Ratia, H., Montesano, L., and Martinez-Cantin, R. (2012). On the performance of maximum likelihood inverse reinforcement learning. Available at: https://arxiv.org/abs/1202.1558.

Ratliff, L. J., Dong, R., Ohlsson, H., and Sastry, S. S. (2014). Incentive design and utility learning via energy disaggregation. *IFAC Proceedings Volumes*.

Sayre, G. A. and Ruan, D. (2014). Automatic treatment planning with convex imputing. *Journal of Physics: Conference Series*, 489(1).

Schaefer, A. (2009). Inverse integer programming. *Optimization Letters*, 3(4):483–489.

Simchi-Levi, D. (2014). OM research: From problem-driven to data-driven research. *Manufacturing & Service Operations Management*, 16(1):2–10.

Sokkalingam, P. T., Ahuja, R. K., and Orlin, J. B. (1999). Solving inverse spanning tree problems through network flow techniques. *Operations Research*, 47(2):291–298.

Stabler, B. (2018). Transportation networks. https://github.com/bstabler/TransportationNetworks.

Syed, U. and Schapire, R. E. (2007). A game-theoretic approach to apprenticeship learning. In *Conference on Neural Information Processing System (NIPS)*. Available at: https://papers.nips.cc/paper/3293-a-game-theoretic-approach-to-apprenticeship-learning.

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. (2005). Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning (ICML)*. Available at: https://dl.acm.org/citation.cfm?doid=1102351.1102464.

Thai, J. and Bayen, A. M. (2018). Imputing a variational inequality function or a convex objective function: A robust approach. *Journal of Mathematical Analysis and Applications*, 457(2):1675–1695.

Troutt, M. D., Brandybarry, A. A., Sohn, C., and Tadisina, S. K. (2008). Linear programming system identification: The general nonnegative parameters case. *European Journal on Operational Research*, 185:63–75.

Troutt, M. D., Gwebu, K. L., Wang, J., and Brandyberry, A. A. (2011). Some experiments on subjective optimisation. *International Journal of Operational Research*, 12(1):79–103.

Troutt, M. D., Pang, W.-K., and Hung-Huo, S. (2006). Behavioral estimation of mathematical programming objective function coefficients. *Management Science*, 52(3):422–434.

Troutt, M. D., Tadisina, S. K., Sohn, C., and Brandyberry, A. A. (2005). Linear programming system identification. *European Journal on Operational Research*, 161:663–672.

Vovk, V. G. (1990). Aggregating strategies. In *Conference on Learning Theory (COLT)*. Available at: https://dl.acm.org/citation.cfm?id=92672.

Yang, I., Zeilinger, M. N., and Tomlin, C. J. (2014). Utility learning model predictive control for personal electric loads. In *53rd IEEE Conference on Decision and Control*, pages 4868–4874. Available at:https://ieeexplore.ieee.org/abstract/document/7040149.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. Technical report, School of Computer Science, Carnegie Mellon University. available at: http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/Web/People/maz/publications/techconvex.pdf.