A

On Learning How Players Learn: Estimation of Learning Dynamics in the Routing Game

Walid Krichene, University of California, Berkeley
Mohamed Chedhli Bourguiba, Ecole Polytechnique
Kiet Lam, University of California, Berkeley
Alexandre Bayen, University of California, Berkeley

The routing game models congestion in transportation networks, communication networks, and other cyber physical systems in which agents compete for shared resources. We consider an online learning model of player dynamics: at each iteration, every player chooses a route (or a probability distribution over routes, which corresponds to a flow allocation over the physical network), then the joint decision of all players determines the costs of each path, which are then revealed to the players.

We pose the following estimation problem: given a sequence of player decisions and the corresponding costs, we would like to estimate the parameters of the learning model. We consider in particular entropic mirror descent dynamics and reduce the problem to estimating the learning rates of each player.

In order to demonstrate our methods, we developed a web application that allows players to participate in a distributed, online routing game, and we deployed the application on Amazon Mechanical Turk. When players log in, they are assigned an origin and destination on a shared network. They can choose, at each iteration, a distribution over their available routes, and each player seeks to minimize her own cost. We collect a data set using this platform, then apply the proposed method to estimate the learning rates of each player. We observe in particular that after an exploration phase, the joint decision of the players remains within a small distance of the set of Nash equilibria. We also use the estimated model parameters to predict the flow distribution over routes, and compare our predictions to the actual distributions, showing that the online learning model can be used as a predictive model over short horizons. Finally, we discuss some of the qualitative insights from the experiments, and give directions for future research.

ACM Reference Format:
Walid Krichene, Mohamed Chedhli Bourguiba, Kiet Lam, and Alexandre Bayen, 2016. On Learning How Players Learn: Estimation of Learning Dynamics in The Routing Game *ACM Trans. Cyb. Phys. Syst.* V, N, Article A (January YYYY), 22 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

The routing game is a non-cooperative game that models congestion in many cyber physical systems (CPS) in which non-cooperative agents compete for shared resources, such as transportation networks (the resources being roads) and communication networks (the resources being communication links) [Beckmann et al. 1955; Roughgarden 2007; Ozdaglar and Srikant 2007]. The game is played on a directed graph that represents the network, and each player has a source node and destination node, and seeks to send traffic (either packets in a communication setting, or cars in a transportation

This work is supported by the National Science Foundation, under grant XXX.
Author's addresses: W. Krichene and K. Lam, Electrical Engineering and Computer Sciences department, University of California, Berkeley; M. C. Bourguiba, Ecole Polytechnique; A. Bayen, Civil and Environmental Engineering department, Electrical Engineering and Computer Sciences department, University of California, Berkeley.

DOI: http://dx.doi.org/10.1145/0000000.0000000

setting) while minimizing the total delay of that traffic. The delay is determined by the joint decision of all players, such that whenever an edge has high load, it becomes congested and any traffic using that edge incurs additional delay. This delay is given by a congestion function that models the underlying physical process. This model of congestion is simple yet powerful, and routing games have been studied extensively since the seminal work of Beckman [Beckmann et al. 1955].

1.1. Learning models and convergence to Nash equilibria

The Nash equilibria of the game are simple to characterize, and have been used to quantify the inefficiency of the network [Roughgarden 2007], using the price of anarchy definition due to [Koutsoupias and Papadimitriou 1999]. However, the Nash equilibrium concept may not offer a good descriptive model of actual behavior of players. Besides the assumption of rationality, which can be questioned [Simon 1955], the Nash equilibrium assumes that players have a complete description of the structure of the game, their own cost functions, and those of other players. This model is arguably not very realistic for the routing game, as one does not expect users of a network to have an accurate representation of the cost function on every edge, or of the other users of the network. One alternative model of players is a model of repeated play [Marden and Shamma 2013; Fox and Shamma 2013; Marden et al. 2013], sometimes called learning models [Cesa-Bianchi and Lugosi 2006] or adjustment models [Fudenberg and Levine 1998]. In such models, one assumes that each player makes decisions *iteratively* (instead of playing a one-shot game), and uses the outcome of each iteration to adjust their next decision. Formally, if $x_k^{(t)}$ is the decision of player k at iteration t (in this case a flow distribution over available routes), and $\ell_k^{(t)}$ is the vector of costs (in this case, delays), then player k faces a sequential decision problem in which she iteratively chooses $x_k^{(t)}$ then observes $\ell_k^{(t)}$. These sequential decision problems are coupled through the cost functions, since $\ell_k^{(t)}$ depends not only on $x_k^{(t)}$ but also on $x_{k'}^{(t)}$ for $k' \neq k$. Such models have a long history in game theory, and date back to the work of [Hannan 1957] and [Blackwell 1956]. In recent years, there has been a resurgence of research on the topic of learning in games using sequential decision problems, see for example [Cesa-Bianchi and Lugosi 2006] and references therein.

When designing a model of player decisions, many properties are desirable. Perhaps the most important property is that the dynamics should be consistent with the equilibrium of the game, in the following sense: Asymptotically, the learning dynamics should converge to the equilibrium of the one-shot game (be it Nash equilibrium or other, more general equilibrium concepts). In this sense, players “learn” the equilibrium asymptotically. Much progress has been made in recent years in characterizing classes of learning dynamics which are guaranteed to converge to an equilibrium set [Freund and Schapire 1999; Hart and Mas-Colell 2001; Hart 2005; Fox and Shamma 2013; Marden et al. 2013; Arslan and Shamma 2004]. In particular for the routing game, different models of learning have been studied for example in [Fischer and Vöcking 2004; Blum et al. 2006; Kleinberg et al. 2009; Krichene et al. 2015a; Krichene et al. 2015b], with different convergence guarantees.

1.2. A mirror descent model of learning

We will focus in particular on the mirror descent model used in [Krichene et al. 2015b], since it offers a large family of models that have strong convergence guarantees to Nash equilibria. This model describes the learning dynamics as solving, at each step, a simple minimization problem parameterized by a learning rate. It is described in detail in Section 2.2, but at a high level, the learning model can be thought of as an update algorithm that specifies $x_k^{(t+1)}$ as a function of $x_k^{(t)}$, $\ell_k^{(t)}$ and a learning rate $\eta_k^{(t)}$. The learning rate intuitively trades-off two terms: The first term encourages allocating the flow to the best routes of the previous iteration, and the second term penalizes large deviations between $x_k^{(t)}$ and $x_k^{(t+1)}$ (thus encourages *stationarity* of the sequence of decisions). Therefore the learning rate describes how aggressive the update is: A small learning rate results in a small change in strategy, while a large learning rate results in a significant change.

1.3. Estimating the learning rates

Motivated by this interpretation of the learning dynamics, we propose the following estimation problem: Given a sequence of observed player decisions $(\bar{x}_k^{(t)})$, and the sequence of corresponding costs $(\bar{\ell}_k^{(t)})$, can we estimate the parameters of the learning model to fit these observations? These quantities are effectively measured in our experimental setting using the routing game web application, and can be measured on transportation networks using many existing traffic monitoring and forecasting systems, such as the Mobile Millennium system [Bayen et al. 2011] or the Grenoble Traffic Lab [Canudas De Wit et al. 2015].

More precisely, we assume that the player is using a given learning algorithm with an unknown sequence of learning rates $(\eta_k^{(t)})$, and we estimate the learning rates given the observations $(\bar{x}_k^{(t)})$, $(\bar{\ell}_k^{(t)})$. One way to pose the estimation problem is to minimize, at each iteration, the distance between the prediction of the model $x_k^{(t+1)}(\eta)$, and the actual decision $\bar{x}_k^{(t+1)}$. We show in particular that for a careful choice of the distance function, this problem is convex in η and can be solved efficiently. This method allows us to estimate one parameter $\eta_k^{(t)}$ per iteration t and per player k. When we have a sequence of observations available, it can be desirable to control the complexity of the model by assuming a parameterized sequence of learning rates, instead of estimating each term separately. Thus, we propose a second method which assumes that the learning rate is of the form $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$, with parameters $\eta_k^{(0)} > 0$ and $\alpha_k \in (0, 1)$. The resulting estimation problem is non-convex in general, but since it is a two dimensional problem (only two parameters to estimate), it can be minimized efficiently.

1.4. Summary of contributions and organization of the article

Our main contributions are to

(1) Pose the learning rate estimation problem, and show that it is convex for an appropriate choice of the distance function. We also give an example application of the estimated model: It can be used to predict the decision of the players over the next few iterations, by propagating the model forward with the estimated values of the learning rates.
(2) Give the first online implementation at scale of a routing game and deploy on the Amazon Mechanical Turk platform in order to collect data on routing decisions. We developed a web application in which a master user can create an instance of the routing game by defining a graph and cost functions on edges of the graph. Then other users can connect to the interface as players. The game then proceeds similarly to our learning model: At each iteration, every player chooses a flow distribution on their available routes (using a graphical user interface with sliders), then their decisions are sent to a backend server, which computes the total cost of each route, and sends this information back to the players. We discuss and propose solutions to some technical and experimental challenges associated with such ex-

periments, such as player synchronization, and handling attrition (players losing connection or dropping out of the game).

(3) Apply the proposed methods to the data collected from the routing game system, and give quantitative and qualitative insights into the decision dynamics of human players. In particular, we observed that in the first few iterations, the flow distributions tend to oscillate, which typically corresponds to a high value of estimated learning rates. For later iterations, the flow distributions remain close to equilibrium, and the learning rates are lower, although some players may occasionally move the system away from equilibrium by performing aggressive updates. It was also interesting to observe that in some rare cases, the best fit is obtained for a *negative learning rate*, which means that the player updated her strategy by assigning more traffic to routes with higher cost, a counter-intuitive behavior which is hard to model (we comment on this in Section 5). Finally, we comment on the performance of the prediction over a short horizon, which indicates that the mirror descent model is a good descriptive model for player behavior in this setting.

The remainder of the article is organized as follows: In Section 2, we formally define the routing game and the mirror descent dynamics and review its convergence guarantees. In Section 3, we pose the learning rate estimation problem in the entropy case, then extend it to the generalized entropy case. We also briefly discuss the flow prediction problem. In Section 4, we describe the experimental setting and the nature of the collected data. We also give some implementation details of the web application and how it can be interfaced with the Amazon Mechanical Turk platform. In Section 5, we use the data collected from the experiment to solve the estimation and prediction tasks. We comment on the quality of the prediction, and give some qualitative and quantitative insights into the decision dynamics. We conclude in Section 6 by summarizing our results and giving directions for future research.

2. THE ROUTING GAME AND THE LEARNING MODEL

In this section, we give the definition of the (one-shot) routing game, and the model of learning dynamics.

2.1. The routing game

The routing game is played on a directed graph $\mathcal{G} = (V, E)$, where V is a vertex set and $E \subset V \times V$ is an edge set. The players will be indexed by $k \in \{1, \dots, K\}$, and each player is associated with an origin vertex $o_k \in V$, a destination vertex $d_k \in V$, and a traffic mass $m_k \geq 0$ that represents the total traffic that the player needs to send from o_k to d_k. The set of available paths connecting o_k to d_k will be denoted by $\mathcal{P}_k$, and the action set of player k is simply how to allocate the total mass m_k along paths in $\mathcal{P}_k$. This action set can be described simply by the probability simplex over $\mathcal{P}_k$, which we denote by $\Delta^{\mathcal{P}_k} = \{x_k \in \mathbb{R}_+^{|\mathcal{P}_k|} : \sum_{p \in \mathcal{P}_k} x_{k,p} = 1\}$. In other words, each player k chooses a distribution $x_k \in \Delta^{\mathcal{P}_k}$ over their available paths, so her total flow contribution to a path p is $m_k x_{k,p}$. The joint decision of all players is denoted by $x = (x_1, \dots, x_K)$. The costs of the players are then determined as follows:

a) The cost on an edge e is $c_e(\phi_e(x))$, where $c_e(\cdot)$ is a given, increasing function (this models the actual cost due to the physical process, for example delay on a road segment due to accumulation of cars), and $\phi_e(x)$ is the total traffic flow on edge e induced by x, obtained simply by summing all the path flows that go through that edge, i.e. $\phi_e(x) = \sum_{k=1}^{K} \sum_{\{p \in \mathcal{P}_k : e \in p\}} m_k x_{k,p}$.

b) The cost on a path $p \in \mathcal{P}_k$ is denoted by $\ell_{k,p}(x)$, and is the sum of edge costs along the path, i.e. $\ell_{k,p}(x) = \sum_{e \in p} c_e(\phi_e(x))$.

c) The cost for player k is the total path cost for all the traffic sent by player k, i.e. $\sum_{p\in\mathcal{P}_k} m_k x_{k,p}\ell_{k,p}(x)$. This is simply the inner product between the flow vector $m_k x_k$ and the cost vector $\ell_k(x)$, which we denote by $\langle \ell_k(x), m_k x_k\rangle$.

Remark 2.1 (*A note on the player model*). Some formulations of the routing game, e.g. [Sandholm 2001; Krichene et al. 2015a], define the game in terms of *populations* of players, such that each population is an infinite set of players with the same origin and destination. This assumes that each player contributes an infinitesimal amount of flow, so each player can play a single path. In our model, each player is macroscopic, and can split its traffic across multiple routes. Both models are equivalent in terms of analysis, the only difference is the interpretation of the model. We choose the finite player interpretation because it is more consistent with our experimental setting, since we run the game with finitely many players.

Definition 2.2 (*Nash equilibrium*). A distribution $x^\star = (x_1^\star, \dots, x_K^\star)$ is a Nash equilibrium (also called Wardrop equilibrium in the traffic literature, in reference to [Wardrop 1952]) if it satisfies the following condition: For all other feasible distributions $x = (x_1, \dots, x_K)$ and for all k, $\langle \ell_k(x^\star), x_k - x_k^\star\rangle \geq 0$.

In words, $x^\star$ is a Nash equilibrium if for every player k, the expected cost under $x_k^\star$ is lower than the expected cost under any other distribution x_k (thus player k has no incentive to unilaterally change her distribution x_k). If we define the inner product $\langle x, \ell\rangle = \sum_k \langle x_k, \ell_k\rangle$, then this is equivalent to: $x^\star$ is an equilibrium if and only if $\langle \ell(x^\star), x - x^\star\rangle \geq 0$ for all feasible x. This variational inequality is, in fact, equivalent to the first-order optimality condition of the following potential function, usually referred to as the Rosenthal potential, in reference to [Rosenthal 1973]:

PROPOSITION 2.3 (EXISTENCE OF A CONVEX POTENTIAL). *Consider a routing game with increasing edge costs $c_e(\cdot)$, and define the following function*

$$f(x) = \sum_{e\in E} \int_0^{\phi_e(x)} c_e(u)du.$$

Then f is convex its gradient is $\nabla f(x) = \ell(x)$.

This result can be found for example in [Roughgarden 2007]. Due to the fact that the delay function $\ell(\cdot)$ coincides with the gradient field $\nabla f(\cdot)$ of the Rosenthal potential, the Nash condition can be rewritten as $\langle \nabla f(x^\star), x - x^\star\rangle \geq 0$ for all feasible x, and since f is convex, this is a necessary and sufficient condition for optimality of $x^\star$ (see e.g. Section 4.2.3 in [Boyd and Vandenberghe 2010]). Therefore the set of Nash equilibria is exactly the set of minimizers of the convex potential f. This is important both for computation (computing a Nash equilibrium can be done by minimizing a convex function), and for modeling: One can model player dynamics as performing a distributed minimization of the potential function. More precisely, if we adopt the point of view presented in the introduction, in which each player faces a sequential decision problem, and plays $x_k^{(t)}$ then observes $\ell_k(x^{(t)})$, then this corresponds to a first-order distributed optimization of the function f, where each player is responsible for updating the variables $x_k^{(t)}$, and observes, at each iteration, the gradient $\ell_k(x^{(t)}) = \nabla_{x_k} f(x^{(t)})$. Using this connection to distributed optimization, a model of player dynamics was proposed in [Krichene et al. 2015b]. We review the model in the next Section.

2.2. The learning model: Mirror descent dynamics

Each player is assumed to perform a mirror descent update given by the following algorithm:

Algorithm 1 Distributed mirror descent dynamics with DGF ψ_k and learning rates $(\eta_k^{(t)})$.

```
1: for each iteration $t \in \{1, 2, \dots\}$ do
2:     for each player $k \in \{1, \dots, K\}$ do
3:         Play $x_k^{(t)}$,
4:         Observe $\ell_k^{(t)} = \nabla_{x_k} f(x^{(t)})$,
5:         Update distribution by solving the problem
               $x_k^{(t+1)} = \arg\min_{x_k \in \Delta^{\mathcal{P}_k}} \left[\eta_k^{(t)} \left\langle \ell_k(x^{(t)}), x_k \right\rangle + D_{\psi_k}(x_k, x_k^{(t)})\right]$    (1)
```

In the update equation (1), $D_{\psi_k}(x_k, x_k^{(t)})$ is the Bregman divergence between the distributions x_k and $x_k^{(t)}$, defined as $D_\psi(x, y) = \psi(x) - \psi(y) - \langle \nabla\psi(y), x - y\rangle$, for a strongly convex function ψ, called the distance generating function (DGF). In particular, $D_\psi(x, y)$ is non-negative, and it is zero if and only if $x = y$ (by strong convexity of ψ). For a review of Bregman divergences and their uses in optimization, see for example [Censor and Zenios 1997; Banerjee et al. 2005]. Some special cases include:

a) The Euclidean case: If $\psi(x) = \frac{\|x\|_2^2}{2}$, then $D_\psi(x, y) = \frac{\|x-y\|_2^2}{2}$. In this case, mirror descent reduces to the projected gradient descent algorithm.
b) The entropy case: If $\psi(x) = -H(x)$ where $H(x) = -\sum_p x_p \ln x_p$ is the entropy, then $D_\psi(x, y) = \sum_p x_p \ln \frac{x_p}{y_p}$ is the Kullback-Leibler (KL) divergence from x to y. In this case, the mirror descent algorithm is sometimes called the entropic descent [Beck and Teboulle 2003], or exponentiated gradient descent [Kivinen and Warmuth 1997].

Mirror descent is a general method for convex optimization proposed in [Nemirovsky and Yudin 1983]. The model in Algorithm 1 is a distributed version of mirror descent, applied to the Rosenthal potential function f (defined in Proposition 2.3). To give some intuition of the method, the first term in the minimization problem (1), $\left\langle \ell_k^{(t)}, x_k \right\rangle$, can be thought of as a linear approximation of the potential function (since $\ell(x) = \nabla f(x)$), and the second term $D_\psi(x_k, x_k^{(t)})$ penalizes deviations from the previous iterate $x_k^{(t)}$. This the algorithm minimizes a linear combination of the two terms, and the learning rate $\eta_k^{(t)}$ determines the tradeoff between them, and can be thought of as a generalized step size: A smaller $\eta_k^{(t)}$ results in a distribution which is closer to the current $x_k^{(t)}$. Thus, from the potential function point of view, the player minimizes a linearization of the potential plus a Bregman divergence term that keeps x_k close to $x_k^{(t)}$. From the routing game point of view, minimizing the first term $\left\langle \ell_k^{(t)}, x_k \right\rangle$ encourages putting weight on the paths that have smaller cost during the previous iteration, and the second term keeps the distribution close to its current value; the learning rate parameter $\eta_k^{(t)}$ determines how aggressive the player is in shifting traffic to the paths which had a lower cost during the previous iteration.

The convergence of this distributed learning model is discussed in [Krichene et al. 2015b]. The learning dynamics given in Algorithm 1 is guaranteed to converge under the following assumptions:

THEOREM 2.4 (THEOREM 3 IN [KRICHENE ET AL. 2015B]). *Consider the routing game with mirror descent dynamics defined in Algorithm 1, and suppose that for all k, $\eta_k^{(t)}$ is decreasing to 0. Then $f(x^{(t)}) - f(x^\star) = \mathcal{O}\left(\sum_k \left(\frac{1}{t\eta_k^{(t)}} + \frac{\sum_{\tau=1}^t \eta_k^{(\tau)}}{t}\right)\right)$.*

In particular, if the learning rates are polynomially decreasing, $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$, with $\alpha_k \in (0,1)$, then one can bound the sum

$$\sum_{\tau=1}^{t} \eta_k^{(t)} = \eta_k^{(0)} \sum_{\tau=1}^{t} \tau^{-\alpha_k} \le \eta_k^{(0)} \int_0^t \tau^{-\alpha_k} d\tau = \frac{\eta_k^{(0)}}{1-\alpha_k} t^{1-\alpha_k},$$

and, as a consequence,

$$f(x^{(t)}) - f(x^\star) = \mathcal{O}\left(\sum_k t^{\alpha_k - 1}\right) + \mathcal{O}\left(\sum_k t^{-\alpha_k}\right) = \mathcal{O}\left(\sum_k t^{-\min(\alpha_k, 1-\alpha_k)}\right),$$

which converges to 0. While this specific convergence rate does not matter for the purposes of the estimation problem, this convergence guarantee motivates the modeling assumptions we make in the next section: In particular, we will assume that the players use a polynomially decaying sequence of learning rates of the form $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$

3. LEARNING MODEL ESTIMATION

In this section, we assume that we have access to a sequence of observations of traffic distributions $(\bar{x}_k^{(t)})$, and a sequence of delay vectors $(\bar{\ell}_k^{(t)})$, for a given player k. The over bar is used to make a clear distinction between quantities which are observed (e.g. $\bar{x}_k^{(t)}$) and quantities which are estimated or predicted by the model (e.g. $x_k^{(t)}$). Given this sequence of observations, we would like to fit a model of learning dynamics. From the previous section, the learning model in Algorithm 1 is naturally parameterized by the DGF ψ_k and the learning rate sequence $(\eta_k^{(t)})$. We will assume that the DGF is given, and discuss how one can estimate the learning rates.

3.1. Estimating a single term of the learning rates sequence

Given the current flow distribution $\bar{x}_k^{(t)}$ and the current delay vector $\bar{\ell}_k^{(t)} = \ell_k(\bar{x}^{(t)})$, the mirror descent model of Algorithm 1 prescribes that the next distribution is given by

$$x_k^{(t+1)}(\eta) := \underset{x_k \in \Delta^{\mathcal{P}_k}}{\arg\min}\, \eta \left\langle \bar{\ell}_k^{(t)}, x_k \right\rangle + D_{\psi_k}(x_k, \bar{x}_k^{(t)}), \tag{2}$$

where ψ_k is given. Therefore, $x_k^{(t+1)}$ can be viewed as a function of η, (hence the notation $x_k^{(t+1)}(\eta)$) and to estimate η, one can minimize the deviation between what the model predicts and what is observed, as measured by the Bregman divergence; i.e. minimize

$$d_k^{(t)}(\eta) := D_{\psi_k}(\bar{x}_k^{(t+1)}, x_k^{(t+1)}(\eta)). \tag{3}$$

The estimate of the learning rate is then

$$\eta_k^{(t)} = \underset{\eta \ge 0}{\arg\min}\, d_k^{(t)}(\eta). \tag{4}$$

Note that we impose the constraint that $\eta \ge 0$. This is an assumption of the model, and in our experiments, this turns out to be an important constraint, as we will see that $d_k^{(t)}(\eta)$ can, in some rare cases, be minimal for negative values of η if the problem were solved without the non-negativity constraint. This is further discussed in Section 5.

In the next theorem, we show that problem (4) is convex when the DGF is the negative entropy. In fact, one can explicitly compute the gradient of $d_k(\eta)$ in this case, which makes it possible to solve Problem (4) efficiently using gradient descent for example. The negative entropy is a natural choice of DGF for many reasons, both theoretical (it yields an optimal dependence of the convergence rate on the dimension of the problem) and practical (it yields a closed form solution of the update equation (1)), see [Ben-Tal et al. 2001; Beck and Teboulle 2003] for a more detailed discussion.

THEOREM 3.1. *If ψ_k is the negative entropy, then $d_k^{(t)}(\eta) := D_{\psi_k}(\bar{x}_k^{(t+1)}, x_k^{(t+1)}(\eta))$ is a convex function of η, and its gradient with respect to η is given by*

$$\frac{d}{d\eta} d_k^{(t)}(\eta) = \left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} - x_k^{(t+1)}(\eta) \right\rangle.$$

PROOF. When ψ_k is the negative entropy, the solution of the mirror descent update (2) is given by

$$x_{k,p}^{(t+1)}(\eta) = \frac{\bar{x}_{k,p}^{(t)} e^{-\eta \bar{\ell}_{k,p}^{(t)}}}{Z_k^{(t)}(\eta)} \tag{5}$$

where $Z_k^{(t)}(\eta)$ is a normalization constant, given by $Z_k^{(t)}(\eta) = \sum_p \bar{x}_{k,p}^{(t)} e^{-\eta \bar{\ell}_{k,p}^{(t)}}$, see for example [Beck and Teboulle 2003]. Given this expression of $x_k^{(t+1)}(\eta)$, we can explicitly compute the Bregman divergence (which, in this case, is the Kullback-Leibler divergence):

$$\begin{aligned}
d_k(\eta) &= D_{\mathrm{KL}}(\bar{x}_k^{(t+1)}, x_k^{(t+1)}(\eta)) \\
&= \sum_{p \in \mathcal{P}_k} \bar{x}_{k,p}^{(t+1)} \ln \frac{\bar{x}_{k,p}^{(t+1)}}{x_{k,p}^{(t+1)}(\eta)} \\
&= \sum_{p \in \mathcal{P}_k} \bar{x}_{k,p}^{(t+1)} \left(\ln \frac{\bar{x}_{k,p}^{(t+1)}}{\bar{x}_{k,p}^{(t)}} + \eta \bar{\ell}_{k,p}^{(t)} + \ln Z_k^{(t)}(\eta) \right) \\
&= D_{\mathrm{KL}}(\bar{x}_k^{(t+1)}, \bar{x}_k^{(t)}) + \eta \left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} \right\rangle + \ln Z_k^{(t)}(\eta),
\end{aligned} \tag{6}$$

where we used the explicit form (5) of $x_k^{(t+1)}(\eta)$ in the third equality, and the fact that $\sum_p \bar{x}_{k,p}^{(t+1)} = 1$ in the last equality. In this expression, the first term does not depend on η, the second term is linear in η, and the last term is the function $\eta \mapsto \ln Z_k^{(t)}(\eta) = \ln \sum_p \bar{x}_{k,p}^{(t)} e^{-\eta \bar{\ell}_{k,p}^{(t)}}$, which is known to be convex in η (see for example Section 3.1.5 in [Boyd and Vandenberghe 2010]). Therefore $d_k^{(t)}(\eta)$ is convex, and its gradient can be obtained by differentiating each term

$$\begin{aligned}
\frac{d}{d\eta} d_k^{(t)}(\eta) &= \left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} \right\rangle + \frac{\frac{d}{d\eta} Z_k^{(t)}(\eta)}{Z_k^{(t)}(\eta)} \\
&= \left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} \right\rangle + \frac{\sum_p -\bar{\ell}_{k,p}^{(t)} \bar{x}_{k,p}^{(t)} e^{-\eta \bar{\ell}_{k,p}^{(t)}}}{Z_k^{(t)}(\eta)} \\
&= \left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} \right\rangle - \left\langle \bar{\ell}_k^{(t)}, x_k^{(t+1)}(\eta) \right\rangle,
\end{aligned}$$

which proves the claim. □

3.2. Generalized negative entropy

In this section, we propose to use a generalization of the entropy DGF, motivated by the following observation: according to the entropy update and its explicit solution (7), the support of $x_k^{(t+1)}(\eta)$ always coincides with the support of $\bar{x}_k^{(t)}$ (due to the multiplicative form of the solution). As a consequence, if we observe two consecutive terms $\bar{x}_k^{(t)}, \bar{x}_k^{(t+1)}$ such that some p is in the support of $\bar{x}_k^{(t+1)}$ but not in the support of $\bar{x}_k^{(t)}$, the KL divergence $D_{\mathrm{KL}}(\bar{x}_k^{(t+1)}, x_k^{(t+1)}(\eta))$ is infinite for all η, since $\mathrm{support}(\bar{x}^{(t+1)}) \not\subset \mathrm{support}(x_k^{(t+1)}(\eta))$ (in measure theoretic terms, $\bar{x}^{(t+1)}$ is not absolutely continuous with respect to $x_k^{(t+1)}(\eta)$). This is problematic, as the estimation problem is ill-posed in such cases (which do occur in the data set used in Section 5). To solve this problem, we propose two possible approaches:

(1) First, we observe that from equation (6), the KL divergence can be decomposed into two terms as follows:

$$d_k^{(t)}(\eta) = D_{\mathrm{KL}}(\bar{x}_k^{(t+1)}, \bar{x}_k^{(t)}) + \eta \left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} \right\rangle + \ln Z_k^{(t)}(\eta),$$

where the first term, $D_{\mathrm{KL}}(\bar{x}_k^{(t+1)}, \bar{x}_k^{(t)})$ may be infinite (if $\mathrm{support}(\bar{x}^{(t+1)}) \not\subset \mathrm{support}(\bar{x}_k^{(t)})$), but does not depend on η, while the second term, $\eta \left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} \right\rangle + \ln Z_k^{(t)}(\eta)$ is finite for all values of $\eta \geq 0$, regardless of the supports of the observations. Thus, instead of minimizing $d_k^{(t)}(\eta)$, we can minimize $d_k^{(t)}(\eta) - D_{\mathrm{KL}}(\bar{x}_k^{(t+1)}, \bar{x}_k^{(t)})$ and the problem becomes well-posed function regardless of the supports.

(2) Second, instead of using the negative entropy as a DGF, we can consider the following DGF introduced in [Krichene et al. 2015c]: Fix $\epsilon > 0$, and let

$$\psi_\epsilon(x_k) = -H(x + \epsilon) = \sum_p (x_{k,p} + \epsilon) \ln(x_{k,p} + \epsilon).$$

The corresponding Bregman divergence is

$$D_{\psi_\epsilon}(x_k, y_k) = \sum_p (x_{k,p} + \epsilon) \ln \frac{x_{k,p} + \epsilon}{y_{k,p} + \epsilon},$$

and can be interpreted as a generalized KL divergence: it measures the KL divergence between the vectors $x_k + \epsilon$ and $y_k + \epsilon$. In particular, for any $\epsilon > 0$, this Bregman divergence is finite for any $x_k, y_k \in \Delta^{\mathcal{P}_k}$, regardless of their supports, unlike the KL divergence. Finally, it is worth observing that when $\epsilon > 0$, the update equation (1) does not have a closed-form expression as in (5); however, the solution can be computed efficiently using the algorithm of [Krichene et al. 2015c]. In our numerical simulations in Section 5, we use the generalized entropy DGF proposed here.

3.3. Estimating the decay of the learning rate sequence

In the previous section, we proposed a method to estimate a single term of the learning rate sequence. One can of course repeat this procedure at every iteration, thus generating a sequence of estimated learning rates. However, the resulting sequence may not be decreasing. In order to be consistent with the assumptions of the model in Section 2.2, we can assume a parameterized sequence of learning rates (which is by construction decreasing), then estimate the parameters of the sequence, given the observations. Motivated by Theorem 2.4, we will assume, in this section, that $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$ with parameters $\eta_k^{(0)} > 0$ and $\alpha_k \in (0, 1)$.

Given the observations $(\bar{x}_k^{(t)})$ and $(\bar{\ell}_k^{(t)})$, we can define a cumulative cost, $D_k^{(t)}(\alpha_k, \eta_k^{(0)}) := \sum_{\tau=1}^t d_k^{(\tau)}(\eta_k^{(0)} \tau^{-\alpha_k})$, where each term of the sum is as defined in equation (3), then estimate $(\alpha_k, \eta_k^{(0)})$ by solving the problem

$$(\alpha_k, \eta_k^{(0)}) = \underset{\alpha_k \in (0,1), \eta_k^{(0)} \geq 0}{\arg\min} \; D_k^{(t)}(\alpha_k, \eta_k^{(0)}). \tag{7}$$

Note that this problem is non-convex in general, however, since it is low-dimensional (two parameters to estimate), it can also be solved efficiently using non-convex optimization techniques.

3.4. Traffic flow prediction

We discuss one important application of the proposed estimation problem. Once we have estimated the learning rates, we can propagate the model forward in order to predict the distributions of the players for the next time step. More precisely, given the current flow distribution $\bar{x}^{(t)}$ and a current estimate of the learning rate $\eta_k^{(t)}$, according to the learning model in Algorithm 1, the next flow distribution is given by

$$x_k^{(t+1)} = g_k(\bar{x}^{(t)}, \eta_k^{(t)}) := \underset{x_k \in \Delta^{\mathcal{P}_k}}{\arg\min} \, \eta_k^{(t)} \left\langle x_k, \ell_k(\bar{x}^{(t)}) \right\rangle + D_{\psi_k}(x_k, \bar{x}_k^{(t)})$$

where we defined the function g, which takes the current distribution $\bar{x}^{(t)}$ and a learning rate $\eta_k^{(t)}$ and propagates the model forward one step.

We can inductively estimate the next terms by propagating the model further over a horizon h: let $x_k^{(t)} = \bar{x}_k^{(t)}$ and for $i \in \{0, \dots, h-1\}$,

$$x_k^{(t+i+1)} = g_k(x^{(t+i)}, \eta_k^{(t+i)}). \tag{8}$$

Here, we assume that we can extrapolate the learning rate sequence to estimate the terms $\eta_k^{(t+i)}$. If we assume that the learning rate sequence is of the form $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$, then once we have an estimate of $\eta_k^{(0)}$ and α_k, we obtain an estimate of the entire sequence. However, if each term of the sequence is estimated separately, we can simply set the future learning rates to be constant, using one of the following simple methods (which we evaluate in Section 5):

(1) As a baseline method, we simply set $\eta_k^{(t+i)} = \eta_k^{(t-1)}$ for all i (we use the last estimated value).
(2) Second, we set $\eta_k^{(t+i)} = \frac{1}{N} \sum_{n=1}^N \eta_k^{(t-n)}$ for all i (we use the average of the last N estimates, for a fixed parameter N).

We conclude this section by observing that while we chose to apply the model to a simple prediction task, the estimated model can be used, more generally, in any a receding-horizon optimal control problem, by using the current estimate of the model as a plant in the control problem.

4. THE ROUTING GAME WEB APPLICATION

In order to conduct the experiment, we have implemented a web application based on the routing game, using the Python Django Framework. The code for the web application is available on Github at the following url: `www.github.com/walidk/routing`. The application has been deployed on the Heroku service at the following url: `routing-game.herokuapp.com`. We have then interfaced the web application with the Amazon Mechanical Turk platform `www.mturk.com` which allows for easy recruiting of players. In this section, we will describe the architecture of the web application along with how the experiment is conducted on Amazon Mechanical Turk.

4.1. Web Application Architecture

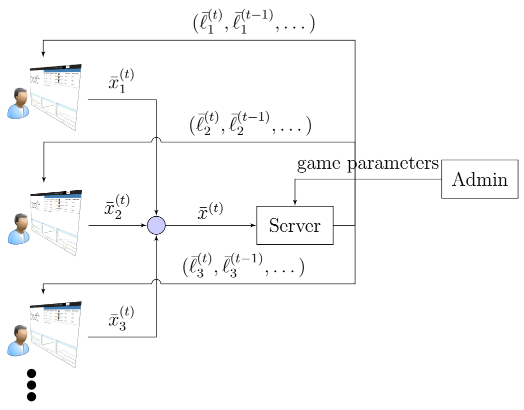

$(\bar{\ell}_1^{(t)}, \bar{\ell}_1^{(t-1)}, \dots)$
$\bar{x}_1^{(t)}$
$(\bar{\ell}_2^{(t)}, \bar{\ell}_2^{(t-1)}, \dots)$
game parameters
Admin
$\bar{x}_2^{(t)}$
$\bar{x}^{(t)}$
Server
$(\bar{\ell}_3^{(t)}, \bar{\ell}_3^{(t-1)}, \dots)$
$\bar{x}_3^{(t)}$

Fig. 1. General architecture of the system. The administrator sets up the game. During iteration t, the clients input the current values of the distributions $\bar{x}_k^{(t)}$ and send them to the server. At the end of the iteration, the server uses these values to compute the cost functions $\bar{\ell}_k^{(t)}$ and sends them back to the clients.

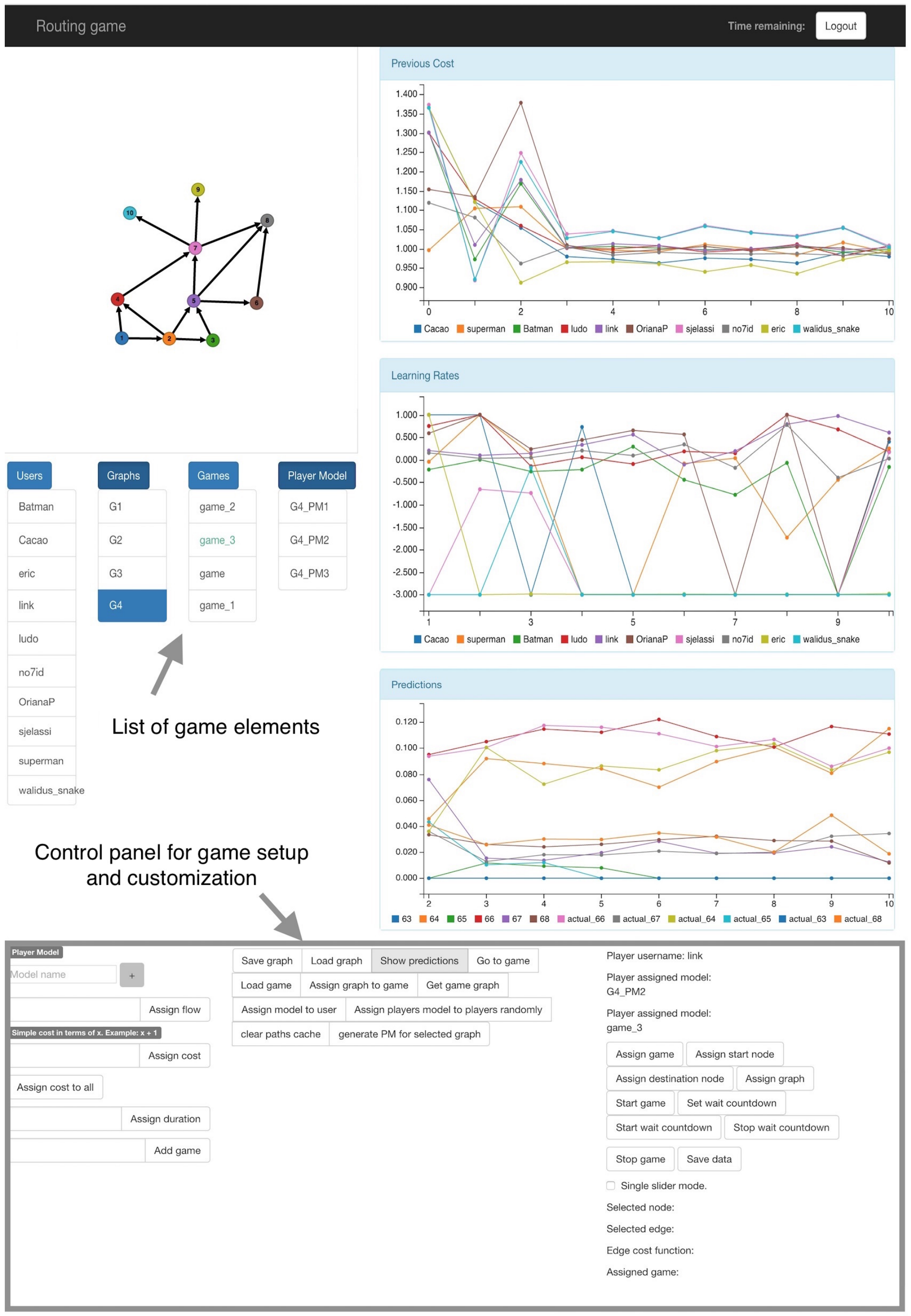
The web application implements the repeated routing game described in Section 2. The general architecture of the system is summarized in Fig. 1. It consists of two different client interfaces, that are used respectively by the administrator of the game and the players, shown in Figures 2 and 3, and a backend server that is responsible for collecting inputs from the clients, updating the state of the game, then broadcasting current information to each player.

4.1.1. Admin Interface. The administrator can set up the game using the admin interface shown in Fig. 2 by:

(1) Creating a graph and defining the cost functions on each edge,
(2) Creating player models. A player model is defined by its origin, destination and total mass. When a player connects to the game, she is randomly assigned to one of the player models,
(3) Setting additional parameters of the game, such as the total number of iterations and the duration of each iteration.

During the game, the administrator can monitor, for each player, her expected cost, learning rate estimates as well as the flow prediction, computed as described in Section 3.4. The administrator can also use the interface to update some of the parameters of the game (such as the duration of each turn), even after the game has started.

4.1.2. Player Interface. Figure 3 shows a screenshot of the client interface for the players. The table is the main element of the graphical user interface, and can be used by the player to set weights on the different paths, using the sliders. The weights determine the flow distribution $\bar{x}_k^{(t+1)}$. The table also shows the previous flow distribution ($\bar{x}_k^{(t)}$), and the previous costs ($\bar{\ell}_k^{(t)}$). Clicking a path on the table will also highlight that path on the graph. The bottom charts show the full history of flows $\bar{x}_k^{(\tau)}$, costs $\bar{\ell}_k^{(\tau)}$, and



Player username: link

Player assigned model:
G4_PM2

Player assigned model:
game_3

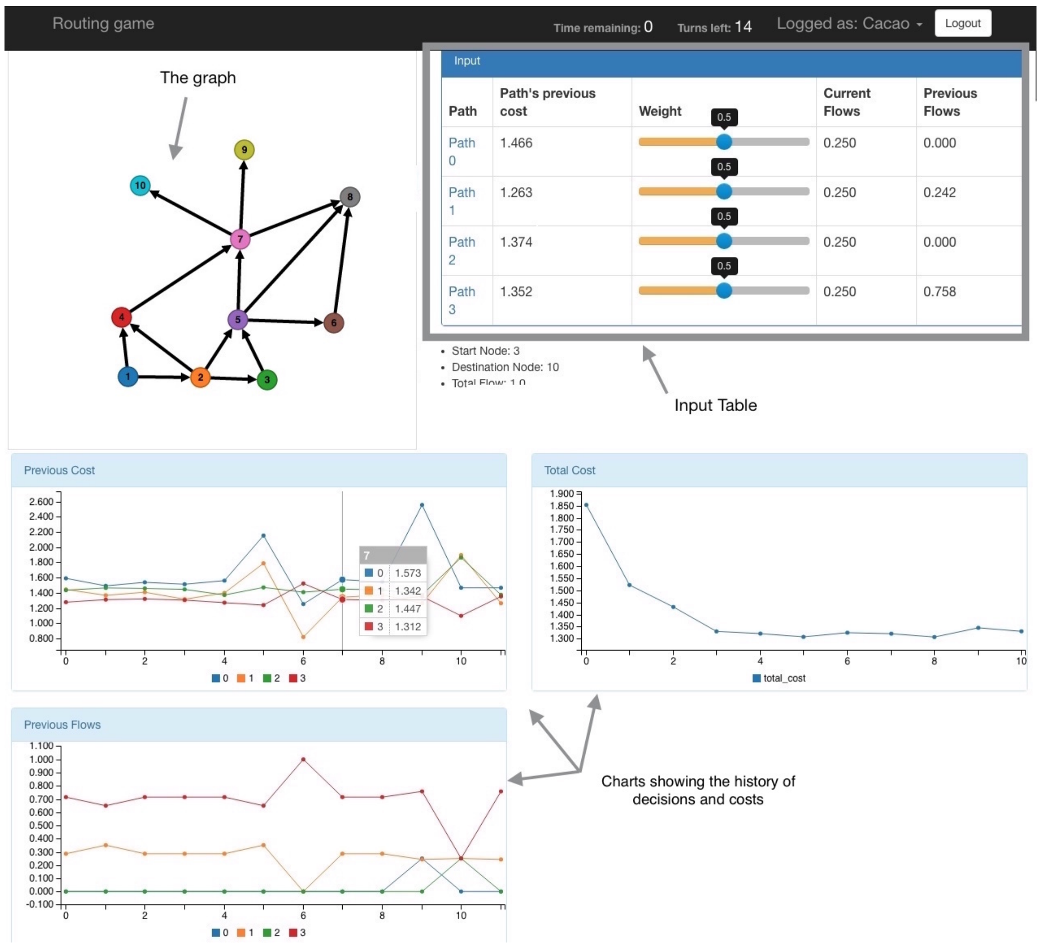
Selected node:

Selected edge:

Edge cost function:

Assigned game:

Fig. 2. Admin interface



Path	Path's previous cost	Weight	Current Flows	Previous Flows
Path 0	1.466	0.5	0.250	0.000
Path 1	1.263	0.5	0.250	0.242
Path 2	1.374	0.5	0.250	0.000
Path 3	1.352	0.5	0.250	0.758

- Start Node: 3
- Destination Node: 10
- Total Flow: 1.0

7	
0	1.573
1	1.342
2	1.447
3	1.312

Fig. 3. User interface

expected costs given by the inner product $\left\langle \bar{x}_k^{(\tau)}, \bar{\ell}_k^{(\tau)} \right\rangle$, for $\tau \leq t$. The origin and destination of the player are displayed on the client interface, underneath the input table. The top navigation bar also shows the time left until the end of the current iteration, and the number of iterations left until the end of the game.

4.1.3. Game Progress. Once a game is set up by the administrator (i.e. the graph of the game is created, the edge costs are set, and the population models are defined), players can log in to the client interface, and each player is assigned to an arbitrary player model (note that several players can share the same model). Once the game starts, it is played in iterations, such that each iteration lasts a specified period of time shown by the timer on top of the client interface (each iteration lasts 30 seconds in our experiments). Each player k can use the sliders to set her flow distribution $x_k^{(t)}$ during iteration t. At the end of the iteration, the server uses the values of $x_k^{(t)}$ for all players $k \in \{1, \dots, K\}$ to compute the costs $\ell_k^{(t)}$, then sends this information to the client side, which then updates the charts and the table with the last value of the cost. Note that each player only has access to the information about her own paths, so in this sense, the learning is completely distributed, as players do not observe the decision or the costs of other players. The decisions of the players $(\bar{x}_k^{(t)})$ and the costs $(\bar{\ell}_k^{(t)})$ are logged by the server, with no identifiable information about the players.

Finally, the application can be set up to run several games consecutively. After the maximum number of iterations is reached, all the players are notified that they will transition to the next game, and they are redirected to a page with a 30 second countdown, at the end of which the next game starts.

4.2. Conducting The Experiment on Amazon Mechanical Turk

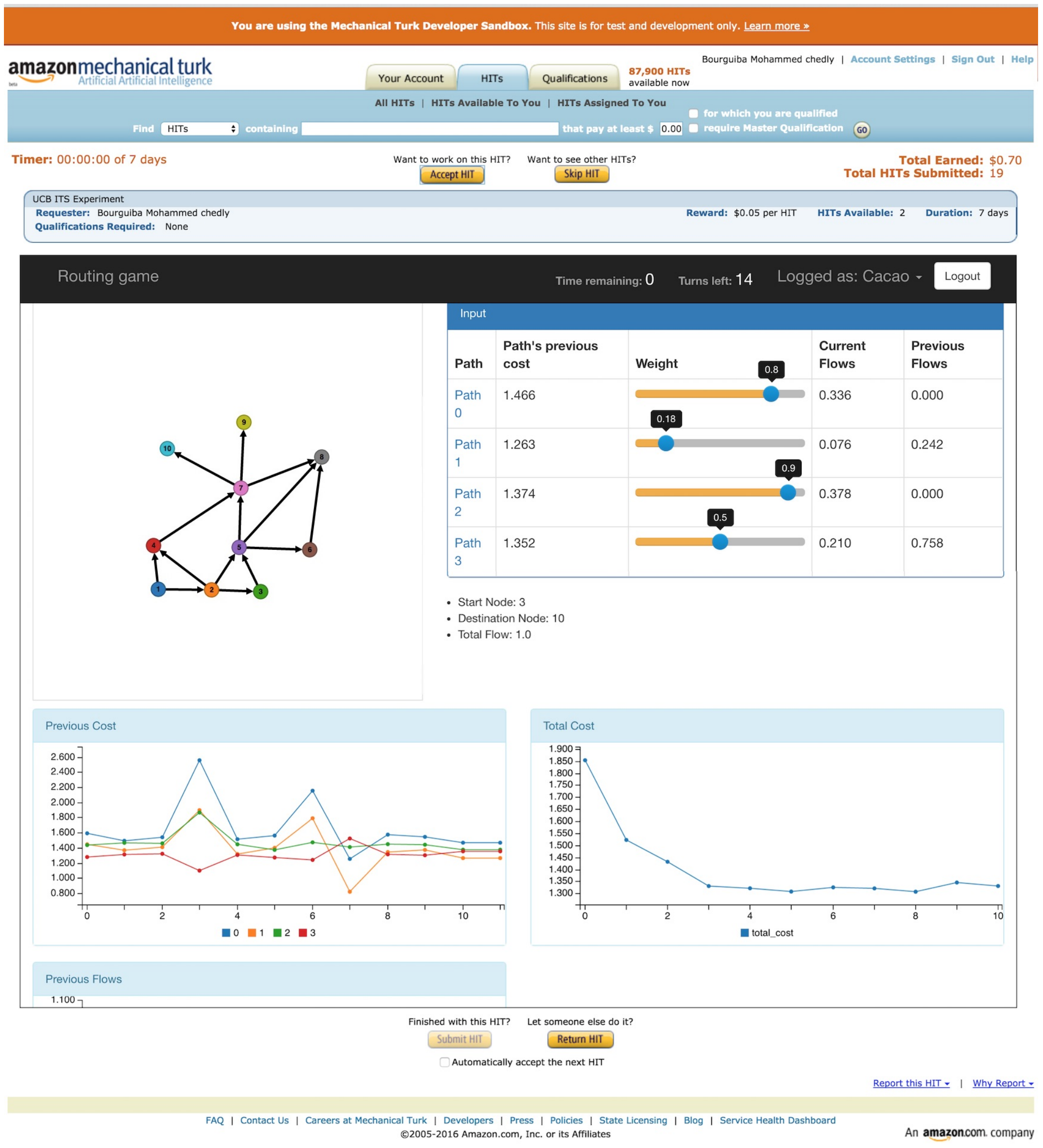
Amazon Mechanical Turk (AMT) is a marketplace for work where it is possible to access on-demand, scalable and cheap workforce. It has been successfully used to perform game theoretic experiments, see [Mason and Suri 2011] and references within. Workers on AMT are paid to perform tasks called *Human Intelligence Tasks* (HITs). Each HIT has a number of assignments. This number defines how many different workers can work on the HIT. By design, HITs are asynchronous tasks. Since the routing game experiment requires simultaneous participation of all players, the corresponding tasks are inherently synchronous. Thus, we followed Mason and Suri's recommendations [Mason and Suri 2011] on how to run a synchronous task on AMT. In particular, the following steps must be achieved:

- Build a panel of potential participants.
- Notify workers.
- Create an online waiting room.
- Handle attrition.

4.2.1. Panel Building and Worker Notification. In order to build a panel of potential participants, a preliminary experiment has been run on AMT, which consisted in a simple, two-player version of the Routing Game. In this version, the first player is controlled by the worker, while the second player is controlled by an algorithm, based on an implementation of the mirror descent model described in Algorithm 1, with a vanishing learning rate $\eta^{(t)} = \frac{1}{\sqrt{t}}$, where t is the iteration number. We refer to this version of the game as the "Worker Vs. AI" Routing Game. It is designed not only to build the panel but also to select players that perform well during the game, i.e that manage to arrive at a distribution that is close to the Nash equilibrium. To measure how close the player is to equilibrium, we simply compute the expected cost of the player $\left\langle \ell_k(x^{(t)}), x_k^{(t)} \right\rangle$, normalized by the cost at equilibrium $\langle \ell_k(x^{\text{Nash}}), x_k^{\text{Nash}} \rangle$. If this value is no more than $1+\epsilon$ for a predefined threshold ϵ (taken to be $.05$ in our experiments), we consider that the player is close to equilibrium. Workers who successfully complete the "Worker Vs. AI" game are then notified by email about the full game, which involves all players simultaneously.

4.2.2. The Waiting Room. During the full game, workers who have been notified may log in and accept the HIT at different times. Thus to ensure that the game starts when a majority of players are present, one needs to implement a waiting room, which allows workers to accept HITs within an interval of a few minutes, then wait for the start of the game. The first worker to arrive to the waiting room triggers a countdown. Once the countdown reaches zero, the game starts with the players that are currently connected. Any player that logs in after the game has started is redirected to a second waiting room, and is allowed to participate in the next game.

4.2.3. Attrition Handling. Despite being synchronous, the game should not stop if one or several players drop out of the game (either by actively abandoning the game, or simply


You are using the Mechanical Turk Developer Sandbox. This site is for test and development only. Learn more »

Bourguiba Mohammed chedly | Account Settings | Sign Out | Help

87,900 HITs available now

Timer: 00:00:00 of 7 days

Total Earned: $0.70
Total HITs Submitted: 19

UCB ITS Experiment
Requester: Bourguiba Mohammed chedly
Qualifications Required: None
Reward: $0.05 per HIT HITs Available: 2 Duration: 7 days

Routing game — Time remaining: 0 Turns left: 14 Logged as: Cacao

Input

Path	Path's previous cost	Weight	Current Flows	Previous Flows
Path 0	1.466	0.8	0.336	0.000
Path 1	1.263	0.18	0.076	0.242
Path 2	1.374	0.9	0.378	0.000
Path 3	1.352	0.5	0.210	0.758

- Start Node: 3
- Destination Node: 10
- Total Flow: 1.0

Your Account
HITs
Qualifications
All HITs | HITs Available To You | HITs Assigned To You
Find HITs containing that pay at least $ 0.00
for which you are qualified
require Master Qualification
GO
Want to work on this HIT?
Accept HIT
Want to see other HITs?
Skip HIT
Logout
Previous Cost
Total Cost
total_cost
Previous Flows
Finished with this HIT?
Submit HIT
Let someone else do it?
Return HIT
Automatically accept the next HIT
Report this HIT | Why Report
FAQ | Contact Us | Careers at Mechanical Turk | Developers | Press | Policies | State Licensing | Blog | Service Health Dashboard
©2005-2016 Amazon.com, Inc. or its Affiliates

Fig. 4. Mechanical Turk interface

due to a connection loss). This phenomenon is referred to as “attrition” in [Mason and Suri 2011]. To handle attrition, we designed the client interface to periodically ping the server, and whenever the server does not receive a ping for more than a given threshold, the player is considered disconnected, and an artificial intelligence takes over the control of that player’s flow distributions, and continues to play in lieu of the player. However, if the player reconnects at a later iteration (i.e. the client interface pings the server again), the A.I. is suspended and the player can resume the game. The A.I. simply implements the mirror descent algorithm described in Algorithm 1.

We considered several options for handling a connection loss:

(1) The player is entirely removed from the game (i.e. the mass of the player is set to zero). This has the negative effect of disrupting the game, since changing the total mass of players will result in a sudden change in path losses. This also results in changing the set of Nash equilibria of the game.
(2) The player is kept in the game but the flow distribution of the player is not updated. While this preserves the total mass and the set of Nash equilibria, it makes it unlikely for the game to converge to a Nash equilibrium (since the disconnected player may be far from an equilibrium).
(3) The player is kept in the game, and the flow distributions of the player are updated by an A.I. that implements the mirror descent model of Algorithm 1. This has the advantage of preserving the total mass and the set of Nash equilibria, and makes it possible for the system to converge to equilibrium even after one or several players have dropped out of the game. This method may affect the conclusions that can be drawn from the experiment, since the experiment does not purely involve human decisions. However, we believe that the advantages of this method outweigh its shortcomings.

4.3. Deploying the routing game application on Heroku

For simple tasks, AMT provides a set of templates that can be used to build HITs. For more complicated tasks such as the routing game (in particular tasks that require synchronization of different workers), AMT offers the possibility of creating external HITs. The requester simply provides a link to the external url of the HIT, and AMT displays this url in a frame as shown in Fig. 4.

Heroku is a cloud platform to deploy web applications. It supports different frameworks such as Python-Django or Java-Play. Heroku also offers different add-ons which we have used in our implementation. For running the experiment with 10 players, we have used the following resources:

— 15 Professional Dynos Standard 1X/2X: Dynos are processes which are used to handle requests. More dynos are required for handling an increased volume of traffic. The number of Dynos should be scaled linearly with the number of players.
— Heroku Postgres Database Hobby Basic Plan.
— Redis Cloud 2.5Gb Plan: Redis is a caching service that we used to cache the values of flow distributions and path costs, in order to minimize database calls.

5. EXPERIMENTAL RESULTS

To illustrate the methods proposed in this article, we ran the experiment on two different networks (shown in Fig. 5), with 10 anonymous players. The first game is played over a horizon of 16 iterations, while the second one is played over a horizon of 25 iterations. In both games, the edge cost functions are taken to be linear increasing. The experiment was conducted according to the protocol described in Section 4.2. We use the data set collected by the experiment to illustrate the estimation and prediction problems proposed in Section 3, and give some comments on some qualitative and quantitative aspects of the decision dynamics of the players.

5.1. Exploration and convergence to equilibrium

First, we evaluate whether the (distributed) decisions of the players converge to the Nash equilibrium of the game. The distance to equilibrium can be measured simply by the Rosenthal potential defined in Proposition 2.3. Fig. 6 shows, for the first game, the potential $f(x^{(t)}) - f(x^\star)$ as a function of iteration t, as well as the corresponding

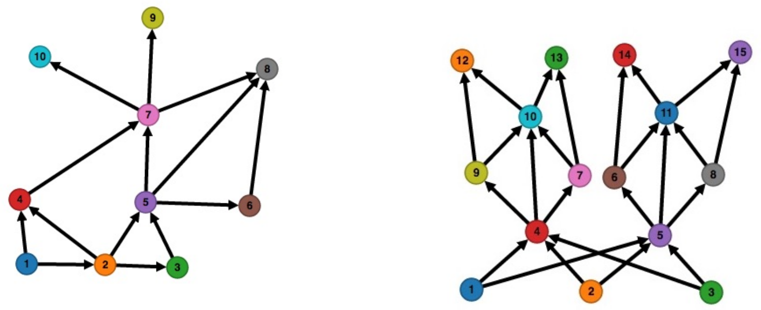


Fig. 5. Graphs used in the two games of the experiment.

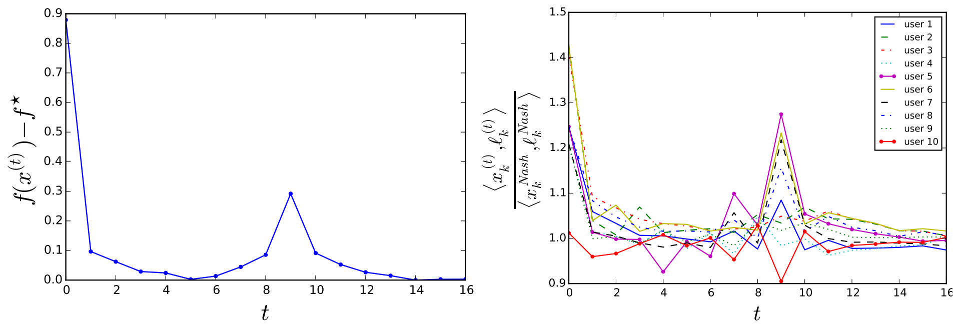


Fig. 6. Exploration and convergence to equilibrium in the first game. The left figure shows the distance to equilibrium, measured by the Rosenthal potential $f(x^{(t)}) - f(x^\star)$ as a function of iteration t, where $x^{(t)} = (x_1^{(t)}, \dots, x_K^{(t)})$ is the joint decision of all players. The right figure shows the costs of each player, normalized by the equilibrium costs $\left\langle x_k^{(t)}, \ell_k^{(t)} \right\rangle / \left\langle x_k^{Nash}, \ell_k^{Nash} \right\rangle$ (so that their values are comparable).

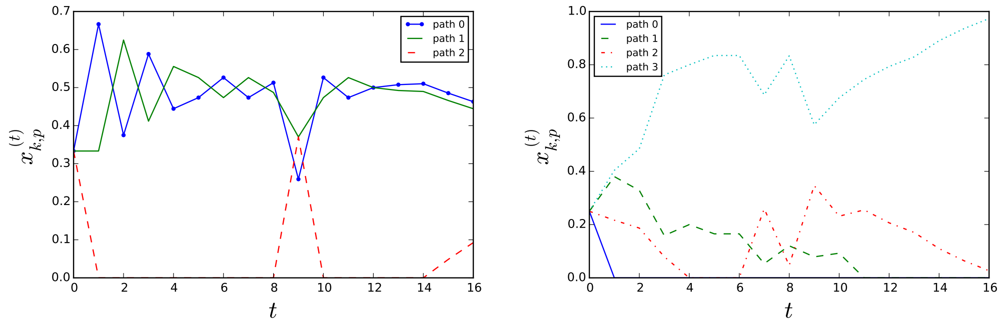


Fig. 7. Sample flow distributions $x_k^{(t)}$ as function of the iteration t for two different players in the first game.

costs $\left\langle x_k^{(t)}, \ell_k^{(t)} \right\rangle$ of the players, normalized by the equilibrium costs $\left\langle x_k^{\text{Nash}}, \ell_k^{\text{Nash}} \right\rangle$ (so that, close to equilibrium, the normalized costs are close to one). Fig. 7 shows the flow distributions $x_k^{(t)}$ for two different players in the first game. We can observe that at the beginning of the game, there is a clear exploration phase in which players tend to make aggressive adjustments in their distributions (observe the oscillations in the early iterations on Fig. 7), while during later turns, the adjustments become, in general, less aggressive and the joint distribution $x^{(t)}$ remains close to equilibrium (as measured by the potential function f on Fig. 6). The system does move away from equilibrium on iteration 9 (due to a player performing an aggressive update), which results in a sharp increase in the potential value, and we can observe that the players react to this sudden change by significantly changing their distribution. Note that this only occurred in the first game, while in the second game no such perturbation was observed. The system quickly recovers after this perturbation, and remains close to equilibrium during the later iterations.

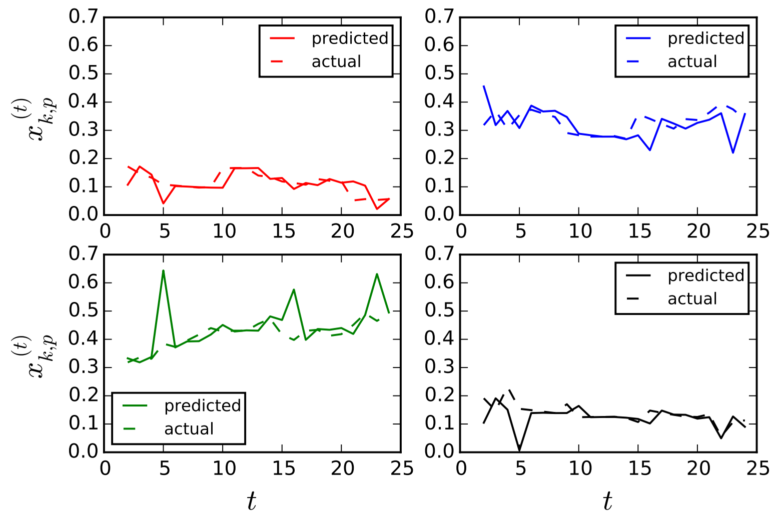


Fig. 8. Comparison of the distributions $x_k^{(t)}$ of the estimated model to the actual distributions $\bar{x}_k^{(t)}$, for player $k = 2$. Each subplot corresponds to a path.

5.2. Estimation and prediction

We now apply the methods proposed in Section 3 to estimate the learning rates of each player, then use the estimated rates to predict the decision of the players over a short horizon. In this section, we take the Bregman divergence to be the generalized entropy defined in Section 3.2, with $\epsilon = 10^{-3}$. We use the data collected during the second game, since it was played over a longer horizon.

First, we solve Problem (4) to estimate the learning rate sequence one term at a time. Fig. 8 compares the estimated distributions by the model, to the actual distributions, for one of the players in the second game. The figure shows that the estimated distributions closely fit the actual distributions, which indicates that the mirror descent model proposed in Section 2 is expressive enough to describe the observed behavior of the players.

In addition to estimating one term of the learning rate sequence at a time, we also compute moving averages of the estimated rates (over a window of 5 iterations), then we use the parameterized form $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$, and estimate $\eta_k^{(0)}$ and α_k by solving problem (7). The results of the three methods are shown in Fig. 9.

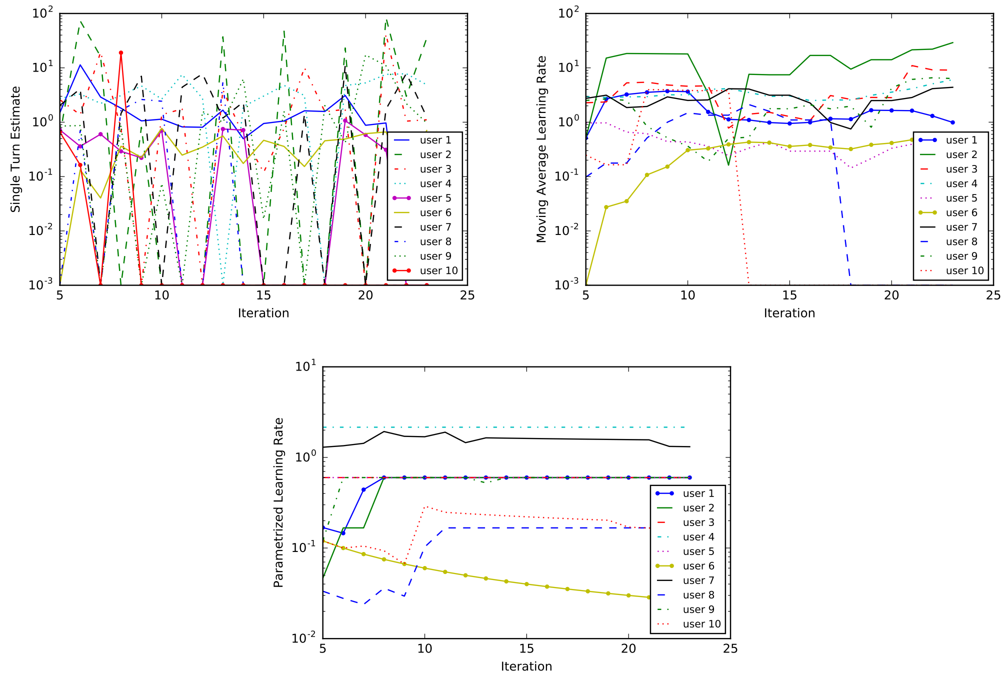
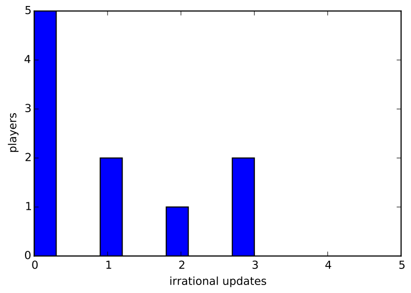


Fig. 9. Estimated sequences of learning rates in semi-logarithmic scale. In the first method, we estimate one term of the sequence at a time, in the second method we use a moving average of the single-term estimates, and in the third method, we use the parameterized form of the estimates, $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$. We start the estimated sequence at $t = 5$ in order to have enough data points to have a good initial estimate.

Irrational updates. It was interesting and perhaps surprising to observe that when estimating learning rates one term at a time, in some rare instances, the objective $d_k^{(t)}(\eta_k^{(t)})$, as defined in equation (4), is minimal at a negative $\eta_k^{(t)}$ (if we ignore the constraint $\eta \geq 0$), which means that the player shifted the probability mass towards paths with *higher* costs. Fig. 10 shows the histogram of the number of irrational updates for the second game. In particular, 50% of the players performed at least one irrational update, and a total of 10 irrational updates were observed across all players (corresponding to 4.17% of total number updates for this game). In the first game, the rate of irrational updates was 6.25%.

Such behavior is hard to interpret or justify (at least within our framework which models players as sequential decision makers). A negative learning rate does not make sense in our model, since the minimization problem (1), which defines the mirror descent update, would encourage shifting mass towards paths with higher cost. Thus we add the constraint $\eta \geq 0$ when solving the estimation problem (4). Note that this problem does not occur when we estimate the entire sequence in its parameterized form, as discussed in Section 3.3.

Predicting future distributions. Next, we use the estimated learning rates to predict the distributions of the players over a short horizon $h \in \{1, \ldots, 7\}$. More precisely, given a horizon h, we compute, at each iteration t, the estimated learning rates up to t, then propagate the model forward from t to $t+h$, by iteratively applying the function g defined in (8). We evaluate each method by computing the average Bregman divergence



Path	$\bar{x}^{(t)}$	$\bar{\ell}^{(t)}$	$\bar{x}^{(t+1)}$
p_1	0.197	2.349	0.251
p_2	0.314	1.856	0.285
p_3	0.266	2.435	0.242
p_4	0.223	2.575	0.222

Fig. 10. Histogram of irrational updates (left), corresponding to iterations t such that the inner product $\left\langle \bar{\ell}_k^{(t)}, \bar{x}_k^{(t+1)} - \bar{x}_k^{(t)} \right\rangle > 0$, which means that the player shifts probability mass to paths with higher costs, which is hard to predict by the model. Example of an irrational update (right), corresponding to iteration $t = 2$ for player P6. In particular, this player decreased the flow on path p_2 even though this is the best path).

(per player and per iteration) between the predicted distribution $x_k^{(t+h)}$ and the actual distribution $\bar{x}_k^{(t+h)}$, i.e.

$$\frac{1}{K}\sum_{k=1}^{K} \frac{1}{t_{\max} - t_{\min}} \sum_{t=t_{\min}}^{t_{\max}-1} D_{\psi_k}(\bar{x}_k^{(t+h)}, x_k^{(t+h)}),$$

where $t_{\min}$ is taken to be equal to 5 (so that there is always a minimal history of observations to estimate the parameters). The results are given in Fig. 11. One can observe that for all methods, as the horizon h increases, the average divergence tends to increase, since the modeling errors propagate and the quality of the predictions degrade. The best overall performance is obtained with the parameterized model $\eta_k^{(t)} = \eta_k^{(0)} t^{-\alpha_k}$, although for $h = 1$, the best prediction is achieved using the per-iteration estimate of $\eta_k^{(t)}$ (since this model has as many parameters as time steps, it allows for a much better fit of the observed data, but has poor generalization performance, i.e. its prediction quickly degrades beyond the first iteration).

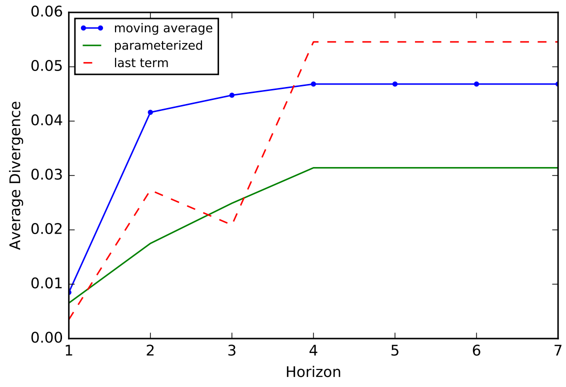


Fig. 11. Average Bregman divergence per player and per iteration, between the predicted distributions and the actual distributions, as a function of the prediction horizon.

6. CONCLUSION

We proposed a problem of model estimation in the routing game, to fit the parameters of a distributed learning model to observations of player decisions. The estimated model can then be used to predict the decisions at future iterations, or, more generally, as a plant model in an optimal control problem.

We considered in particular a model based on the mirror descent algorithm, parameterized by a sequence of learning rates $(\eta_k^{(t)})$, and gave an intuitive interpretation of how this model can describe player behavior. We showed that the problem of estimating one term of the learning rate sequence is convex in the case of the KL divergence (it remains open to prove this result for other Bregman divergences). To control the complexity of the model and to make the estimation consistent with the theoretical assumptions (decreasing learning rates), we proposed to parameterize the sequence with an initial term $\eta_k^{(0)}$ and a decay rate $\alpha_k \in (0, 1)$.

This estimation problem can be extended to estimate the DGF in addition to the learning rates. One way to pose the problem is to consider a finite collection of distance generating functions $\{\psi_i\}_{i\in\mathcal{I}}$, then to assume that each player k uses a DGF that is a linear combination $\psi = \sum_i \theta_{k,i}\psi_i$, then estimate the parameter vector θ_k. This would increase the expressive power of the model.

When we tested these methods on data collected from our routing game experiment, the parameterized sequence estimation outperformed the other methods on the prediction task. Our test results suggest that the mirror descent model can be a good descriptive model of player behavior, although in some rare cases, a player decision can be hard to model (e.g. when a player increases flows on previously bad routes).

REFERENCES

G. Arslan and J.S. Shamma. 2004. Distributed convergence to Nash equilibria with local utility measurements. In *43rd IEEE Conference on Decision and Control*, Vol. 2. 1538–1543 Vol.2.

Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. 2005. Clustering with Bregman Divergences. *J. Mach. Learn. Res.* 6 (Dec. 2005), 1705–1749.

A.M. Bayen, J. Butler, A.D. Patire, CCIT, UC Berkeley ITS, and California Dpartment of Transportation, Division of Research and Innovation. 2011. *Mobile Millennium Final Report*.

Amir Beck and Marc Teboulle. 2003. Mirror Descent and Nonlinear Projected Subgradient Methods for Convex Optimization. *Oper. Res. Lett.* 31, 3 (May 2003), 167–175.

Martin J Beckmann, Charles B McGuire, and Christopher B Winsten. 1955. Studies in the Economics of Transportation. (1955).

Aharon Ben-Tal, Tamar Margalit, and Arkadi Nemirovski. 2001. The Ordered Subsets Mirror Descent Optimization Method with Applications to Tomography. *SIAM J. on Optimization* 12, 1 (Jan. 2001), 79–108.

David Blackwell. 1956. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.* 6, 1 (1956), 1–8.

Avrim Blum, Eyal Even-Dar, and Katrina Ligett. 2006. Routing without regret: on convergence to nash equilibria of regret-minimizing algorithms in routing games. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing (PODC '06)*. ACM, New York, NY, USA, 45–52.

Stephen Boyd and Lieven Vandenberghe. 2010. *Convex Optimization*. Vol. 25. Cambridge University Press.

Carlos Canudas De Wit, Fabio Morbidi, Luis Leon Ojeda, Alain Y. Kibangou, Iker Bellicot, and Pascal Bellemain. 2015. Grenoble Traffic Lab: An experimental platform for advanced traffic monitoring and forecasting. *IEEE Control Systems* 35, 3 (June 2015), 23–39.

Yair Censor and Stavros Zenios. 1997. *Parallel Optimization: Theory, Algorithms and Applications*. Oxford University Press.

Nicolò Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games*. Cambridge University Press.

Simon Fischer and Berthold Vöcking. 2004. On the evolution of selfish routing. In *Algorithms–ESA 2004*. Springer, 323–334.

Michael J. Fox and Jeff S. Shamma. 2013. Population Games, Stable Games, and Passivity. *Games* 4, 4 (2013), 561–583.

Yoav Freund and Robert E Schapire. 1999. Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29, 1 (1999), 79–103.

Drew Fudenberg and David K Levine. 1998. *The theory of learning in games*. Vol. 2. MIT press.

James Hannan. 1957. Approximation to Bayes risk in repeated plays. *Contributions to the Theory of Games* 3 (1957), 97–139.

Sergiu Hart. 2005. Adaptive Heuristics. *Econometrica* 73, 5 (2005), 1401–1430.

Sergiu Hart and Andreu Mas-Colell. 2001. A General Class of Adaptive Strategies. *Journal of Economic Theory* 98, 1 (2001), 26 – 54.

Jyrki Kivinen and Manfred K. Warmuth. 1997. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Information and Computation* 132, 1 (1997), 1 – 63.

Robert Kleinberg, Georgios Piliouras, and Eva Tardos. 2009. Multiplicative updates outperform generic no-regret learning in congestion games. In *Proceedings of the 41st annual ACM symposium on Theory of computing*. ACM, 533–542.

Elias Koutsoupias and Christos Papadimitriou. 1999. Worst-case equilibria. In *In proceedings of the 16th annual symposium on theoretical aspects of computer science*. 404–413.

Walid Krichene, Benjamin Drighès, and Alexandre Bayen. 2015a. Learning Nash equilibria in Congestion Games. *SIAM Journal on Control and Optimization (SICON)* (2015).

Walid Krichene, Syrine Krichene, and Alexandre Bayen. 2015b. Convergence of Mirror Descent dynamics in the Routing Game. In *European Control Conference (ECC)*.

Walid Krichene, Syrine Krichene, and Alexandre Bayen. 2015c. Efficient Bregman Projections onto the Simplex. In *IEEE Conference on Decision and Control, in review*.

J.R. Marden and J.S. Shamma. 2013. Game Theory and Distributed Control. In *Handbook of Game Theory Vol. 4*, H.P. Young and S. Zamir (Eds.). Elsevier Science.

Jason R Marden, Shalom D Ruben, and Lucy Y Pao. 2013. A model-free approach to wind farm control using game theoretic methods. *Control Systems Technology, IEEE Transactions on* 21, 4 (2013), 1207–1214.

Winter Mason and Siddharth Suri. 2011. Conducting behavioral research on Amazon's Mechanical Turk. (2011).

A. S. Nemirovsky and D. B. Yudin. 1983. *Problem complexity and method efficiency in optimization*. Wiley.

Asuman Ozdaglar and R Srikant. 2007. Incentives and pricing in communication networks. *Algorithmic Game Theory* (2007), 571–591.

Robert W Rosenthal. 1973. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory* 2, 1 (1973), 65–67.

Tim Roughgarden. 2007. Routing games. In *Algorithmic game theory*. Cambridge University Press, Chapter 18, 461–486.

William H Sandholm. 2001. Potential games with continuous player sets. *Journal of Economic Theory* 97, 1 (2001), 81–108.

Herbert A. Simon. 1955. A Behavioral Model of Rational Choice. *The Quarterly Journal of Economics* 69, 1 (1955), pp. 99–118.

John Glen Wardrop. 1952. SOME THEORETICAL ASPECTS OF ROAD TRAFFIC RESEARCH.. In *ICE Proceedings: Engineering Divisions*, Vol. 1. Thomas Telford, 325–362.