This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

On Social Optimal Routing Under Selfish Learning

Walid Krichene, *Member, IEEE* Milena Suarez Castillo and Alexandre Bayen *Member, IEEE*

***Abstract*—We consider a repeated routing game over a finite horizon with partial control under selfish response, in which a central authority can control a fraction of the flow and seeks to improve a network-wide objective, while the remaining flow applies an online learning algorithm. This finite horizon control problem is inspired from the one-shot Stackelberg routing game. Our setting is different in that we do not assume that the selfish players play a Nash equilibrium; instead, we assume that they apply an online learning algorithm. This results in an optimal control problem under learning dynamics. We propose different methods for approximately solving this problem: A greedy algorithm and a mirror descent algorithm based on the adjoint method. In particular, we derive the adjoint system equations of the Hedge dynamics and show that they can be solved efficiently. We compare the performance of these methods (in terms of achieved cost and computational complexity) on parallel networks, and on a model of the Los Angeles highway network.**

I. Introduction

ROUTING games provide a game theoretic framework for analyzing the route choices of non-cooperative agents who share a network. They can be used to model transportation or communication networks. Due to the selfishness of individual players (each seeks to minimize her own latency), the resulting equilibrium is, in general, suboptimal from a system-wide perspective. This inefficiency has been quantified using the price of anarchy for example in [1], [2]. Different approaches have been proposed to alleviate this inefficiency, e.g. through congestion pricing [3], capacity allocation [4], or control of a fraction of the flow [5]. The approach of controlling flow on the network is used in communication networks, and is increasingly relevant for transportation networks due to the abundance of routing software and GPS-enabled devices.

In the one-shot routing game, the partial control problem (controlling a fraction of the flow while the remaining flow responds selfishly), is known as Stackelberg routing, and was proved to be NP-hard even in the simple case of parallel networks with linear latencies [5]. Stackelberg equilibria provide a theoretical framework for understanding the inefficiencies of a network and how much they can be alleviated, but they do not capture *route choice dynamics*. In the repeated game, we start by defining the dynamics of the selfish players, in order to formulate a control problem on this dynamical system. We note that other approaches, for example [6], model and control the dynamics of traffic, by using a macroscopic model based on conservation laws, such as the cell transmission model. Our approach is different in that we do not explicitly model the flow dynamics (time scale of minutes or seconds). Instead, we model route choice dynamics (time scale of days). We will consider the Hedge dynamics in particular, perhaps one of the most widely studied algorithms in the online learning literature [8], also known as the multiplicative weight updates [14] in computer science, the entropic descent in the optimization literature [15], and log-linear learning in game theory [20], [21]. Hedge dynamics are known to have vanishing regret, which guarantees that the learning is asymptotically consistent with the Nash equilibrium of the one-shot game, in the following sense (see e.g. [7]): If players follow no-regret dynamics, then asymptotically, their average strategies converge to the set of Nash equilibria of the routing game (i.e. the distance to the set converges to zero). The routing game model and the online learning dynamics are described in detail in Section II.

Given the selfish population dynamics, we formulate, in Section III, a finite horizon optimal control problem: Find the route allocation of the controlled players, given the non-linear dynamics of the selfish players. The non-linear dynamics make this problem non-convex, and an exact solution cannot be computed efficiently in general. We propose several methods to find an approximate solution to this problem in Sections IV and V. The first method is a greedy solution which optimizes one term of the objective function at a time. In the second method, we use the adjoint method [12], [13] to perform a local search using the gradient of the objective function under the non-linear constraints. In particular, we derive the adjoint system equations associated to the Hedge dynamics. We show that the particular structure of our problem makes the adjoint system efficient to solve.

In Section VI, we illustrate the qualitative behavior of these methods on a simple example. We then study the performance and the computational complexity on parallel networks of increasing size. Finally, we perform a test on a model of the Los Angeles highway network in Section VIII, and show the improvement in the total travel time that could be achieved, for various proportions of controlled traffic. In order to evaluate the performance of these methods, we compare the achieved cost to two bounds: An upper-bound given by the social optimum, corresponding to full control (that is, the entire population is controlled, which eliminates the selfish dynamics and makes the problem convex), and a lower bound given by the no-control cost (that is, when the entire population is governed by the Hedge dynamics). Our results indicate that the methods perform well in practice, and scale efficiently in the size of the strategy space (number of routes).

This work was supported in part by FORCES (Foundations Of Resilient CybEr-physical Systems), which receives support from the NSF (award numbers CNS-1238959, CNS-1238962, CNS-1239054, CNS-1239166).

Walid Krichene is with the department of Electrical Engineering and Computer Sciences, 652, Sutardja Dai Hall, UC Berkeley, 94720 CA.

Milena Suarez Castillo is with the Economic Studies Department at INSEE (French national statistical institute).

Alexandre Bayen is with the department of Electrical Engineering and Computer Sciences and the department of Civil and Environmental Engineering, 642, Sutardja Dai Hall, UC Berkeley, 94720 CA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

II. The online selfish routing model

In this section, we review the standard routing game model, which is used for example in [16], [2], and the no-regret routing dynamics used for example in [7], [17], [18].

A. The one-shot routing game

We consider a directed graph $G = (V, E)$, and K populations of players. Population k is described by a source node $s_k \in V$, a destination node $t_k \in V$, and a total demand F_k, which corresponds to the size of the population. Let $\mathcal{P}_k$ be the set of simple paths that connect s_k to t_k. We denote $\mathcal{P}$ the union $\mathcal{P} = \cup_{k=1}^K \mathcal{P}_k$. We assume that different populations have different source-destination pairs (s_k, t_k). As a consequence, the path sets $\mathcal{P}_k$ are disjoint.

1) Path flows and edge loads: A path flow allocation for population k is a vector $f^k \in \mathbb{R}_+^{\mathcal{P}_k}$ such that f_p^k is the flow on path $p \in \mathcal{P}_k$, and $\sum_{p\in\mathcal{P}_k} f_p^k = F_k$. We can write f^k as a scaled distribution vector, $f^k = F_k \mu^k$, where $\mu^k \in \Delta^{\mathcal{P}_k} = \{u \in \mathbb{R}_+^{\mathcal{P}_k} : \sum_{p\in\mathcal{P}_k} u_d = 1\}$, the probability simplex over $\mathcal{P}_k$. We will write

$$f = (f^1, \cdots, f^K) \in F_1\Delta^{\mathcal{P}_1} \times \cdots \times F_K\Delta^{\mathcal{P}_K},$$
$$\mu = (\mu^1, \ldots, \mu^K) \in \Delta^{\mathcal{P}_1} \times \cdots \times \Delta^{\mathcal{P}_K}.$$

The path flow allocation determines the edge loads: for all $e \in E$, the load of edge e is $\phi_e = \sum_{k=1}^K \sum_{p\in\mathcal{P}_k : e\in p} f_p^k$, which can be written in linear form: $\phi_e = (Mf)_e$, where $M \in \mathbb{R}_+^{E\times\mathcal{P}}$ is the incidence matrix of the graph: $\forall e \in E$, $\forall k$ and $\forall p \in \mathcal{P}_k$, $M_{e,p} = 1$ if $e \in p$ and 0 otherwise. We can also write $\phi_e = (\tilde{M}\mu)_e$ where $\tilde{M} \in \mathbb{R}_+^{E\times\mathcal{P}}$ is the weighted incidence matrix: $\forall e \in E$, $\forall k$ and $\forall p \in \mathcal{P}_k$, $\tilde{M}_{e,p} = F_k$ if $e \in p$ and 0 otherwise.

2) Path loss: The load of an edge determines the cost incurred by players utilizing that edge. This cost is called edge latency, or edge loss, and is given by $c_e(\phi_e)$, where c_e is an edge congestion function defined on $\mathbb{R}_+$.

Assumption 1: The edge congestion function c_e is positive increasing, continuously differentiable, and such that $\phi_e \mapsto \phi_e c_e(\phi_e)$ is convex.

The loss incurred on a path $p \in \mathcal{P}_k$ is the sum of edge latencies along the path. We denote ℓ_p^k this loss, so that

$$\ell_p^k(\mu) := \sum_{e\in p} c_e(\phi_e) = \sum_{e\in p} c_e((\tilde{M}\mu)_e).$$

3) Nash equilibria of the routing game: Assuming rationality of the players and perfect information, one can define the Nash equilibria of the game:

Definition 1: A Nash equilibrium of the one-shot routing game is a product distribution $\mu \in \Delta^{\mathcal{P}_1} \times \cdots \times \Delta^{\mathcal{P}_K}$ such that no player has an incentive to unilaterally change her path, i.e.

$$\forall k,\ \text{support}(\mu^k) = \arg\min_{p\in\mathcal{P}_k} \ell_p^k(\mu).$$

We will denote $\mathcal{N}$ the set of Nash equilibria. One can show that $\mathcal{N}$ is the set of solutions to a convex problem (see for example [2]), given by

$$\mathcal{N} = \operatorname*{argmin}_{\mu\in\Delta^{\mathcal{P}_1}\times\cdots\times\Delta^{\mathcal{P}_K}} V(\mu) := \sum_{e\in E} \int_0^{\phi_e = (\tilde{M}\mu)_e} c_e(u)du,$$

where V is a convex function, usually called the Rosenthal potential in reference to [19].

B. Online learning in the repeated routing game

We now assume that the game is played repeatedly in time. Each population maintains a distribution $\mu^{k(t)} \in \Delta^{\mathcal{P}_k}$, where t denotes the iteration number, and faces the following sequential decision problem: At iteration t, the population plays a distribution $\mu^{k(t)}$, then the joint distribution $\mu^{(t)}$ determines the path losses $\ell_p^k(\mu^{(t)})$, which are then revealed to population k. The population can update its distribution $\mu^{k(t+1)}$ given the history of losses. A natural measure of performance for this sequential decision problem is the regret [8], defined as follows: The regret $R^{k(t)}$ of population k is the difference between the expected loss incurred up to t, and the loss of the best fixed strategy in hindsight. That is,

$$R^{k(t)} = \sum_{\tau\le t} \left\langle \mu^{k(\tau)}, \ell^k(\mu^{(\tau)}) \right\rangle - \min_{\mu^k\in\Delta^{\mathcal{P}_k}} \left\langle \mu^k, \sum_{\tau\le t} \ell^k(\mu^{(\tau)}) \right\rangle.$$

The regret is said to be sublinear if $\limsup_{t\to\infty} \frac{R^{k(t)}}{t} \le 0$. If every population has sublinear regret, then the sequence of Cesàro averages of their distributions, $\bar{\mu}^{(t)} := \frac{\sum_{\tau\le t}\mu^{(\tau)}}{t}$, converges to $\mathcal{N}$, see [7], [18], [17]. Here, convergence of $\bar{\mu}^{(t)}$ to the set $\mathcal{N}$ means that $\lim_{t\to\infty} d(\bar{\mu}^{(t)}, \mathcal{N}) = 0$, where $d(\cdot, \mathcal{N})$ is for example the Euclidean distance to the set $\mathcal{N}$. Convergence of $\mu^{(t)}$ is stronger than convergence of $\bar{\mu}^{(t)}$ in general, but it can be guaranteed for a subclass of no-regret algorithms, such as mirror descent algorithms [15] and approximate replicator algorithms [18]. In particular, the Hedge algorithm belongs to both of these families, and we will use it as an example dynamics for the selfish population.

Algorithm 1 Online learning model of the selfish population

1: Given initial distribution $\mu^{k(0)}$ (e.g., uniform distribution)
2: **for** each iteration t **do**
3: Reveal losses $\ell_p^k(\mu^{(t)})$ to population k
4: Update distribution

$$\mu_p^{k(t+1)} = \frac{\mu_p^{k(t)} e^{-\eta_t \ell_p^k(\mu^{(t)})}}{\sum_{q\in\mathcal{P}_k} \mu_q^{k(t)} e^{-\eta_t \ell_q^k(\mu^{(t)})}} \quad (1)$$

The update equation for the Hedge algorithm is given by equation (1), i.e. $\mu_p^{k(t+1)} \propto \mu_p^{k(t)} e^{-\eta_t \ell_p^k(\mu^{(t)})}$, where (η_t) is a sequence of positive learning rates such that $\eta_t \to 0$ and $\sum_t \eta_t = \infty$. In words, the probability of choosing a path decreases, from iteration t to the next, proportionally to the exponential of the loss on that path. The online learning model and the Hedge algorithm are summarized in Algorithm 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

III. Optimal routing subject to selfish learning

Given the dynamics of the selfish populations, we can now define the optimal routing problem. Suppose that a central controller has the task of assigning a fraction of the total flow, and seeks to minimize a network-wide objective function over a finite horizon T. For every population k, let α_k be the fraction of the flow that is assigned by the controller. We will denote $u^{k(t)}$ the vector of path flows assigned by the controller at time step t, so that $u^{k(t)} \in \alpha_k F_k \Delta^{\mathcal{P}_k}$. The fraction of the population which is not controlled is assumed to be selfish, and obeys the online learning model with Hedge updates given in Algorithm 1, starting from a known initial distribution $x^{k(0)}$ and known learning rates (η_t^k) (the learning rates can be estimated from the observed decisions of the selfish populations, as discussed in [22]). We denote $x^{k(t)} \in (1-\alpha_k)F_k\Delta^{\mathcal{P}_k}$ the selfish flow distribution at time t. Finally, we write $u^{(t)} = (u^{1(t)}, \dots, u^{K(t)})$, and use $u^{(1:T)}$ to denote the control variables over the entire horizon, i.e. $u^{(1:T)} = (u^{(t)})_{t\in\{1,\dots,T\}}$. Each vector belongs to a cartesian product of scaled simplices, which we denote $\Delta^u(\alpha, F) = \times_{k=1}^K \alpha_k F_k \Delta^{\mathcal{P}_k}$. We use similar notation for the selfish flow distributions $x^{(t)}$, and use $\Delta^x(\alpha, F) = \times_{k=1}^K (1-\alpha_k)F_k\Delta^{\mathcal{P}_k}$.

Now consider an objective function $J(x,u)$ of the form $\sum_{t=1}^T J^{(t)}(x^{(t)}, u^{(t)})$ with each $J^{(t)}$ convex, and that depends on both the controlled and the selfish flows. The resulting control problem is

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^{T} J^{(t)}(x^{(t)}, u^{(t)}) && (2)\\ \text{subject to} \quad & u^{(t)} \in \Delta^u(\alpha, F),\ t \in \{0,\dots,T\} \\ & x^{(0)} = x^{\text{init}} \\ & x_p^{k(t+1)} = (1-\alpha_k)F_k \frac{x_p^{k(t)} e^{-\eta_t \ell_p^k(x^{(t)}+u^{(t)})}}{\sum_{q\in\mathcal{P}_k} x_q^{k(t)} e^{-\eta_t \ell_q^k(x^{(t)}+u^{(t)})}} && (3) \end{aligned}$$

This problem is non-convex in general, due to the equality constraints (3) corresponding to the learning dynamics. We propose several methods for efficiently finding approximate solutions to this intractable problem.

In each of these methods, we will use the mirror descent algorithm as a minimization method over the set $\Delta^u(\alpha, F)$. Mirror descent is a general method proposed by Nemirovski and Yudin in [23] for solving constrained convex optimization problems. It can be viewed, as observed in [15], as a generalization of projected gradient descent, using a Bregman projection instead of the Euclidean projection, which makes it possible to adapt the projection to the geometry of the feasible set. In particular, when the feasible set is a simplex (or a product of simplices), taking the Bregman projection to be an entropy projection yields an efficient, closed form projection [15]. When minimizing a convex function over the simplex in $\mathbb{R}^n$, the entropic descent method reaches a precision ϵ within $\mathcal{O}(\frac{\log n}{\sqrt{\epsilon}})$ iterations [15]. In the complexity analysis of our methods, we will study the dependence on the dimension n, and not on the precision ϵ. This will provide a good estimate of how the computational cost of each method scales as a function of the problem size (i.e. the number of paths), for a given fixed precision. The mirror descent algorithm is briefly reviewed in the Appendix.

Example 1 (Minimizing total delay): Although the proposed methods apply to general cost functions, we will focus, in our numerical examples, on minimizing total delay on the network, given by

$$J^{(t)}(x,u) = \sum_{k=1}^{K} \sum_{p\in\mathcal{P}_k} (x_p^k + u_p^k)\ell_p^k(x+u). \qquad (4)$$

Here each term $(x_p^k + u_p^k)\ell_p^k(x+u)$ represents the total travel time experienced by the fraction of the population that takes path p.

IV. A greedy method

A. Optimizing terms successively in the cost function

In this method, we minimize the objective function one term at a time, given the state on the previous time steps. That is, we minimize $J^{(t)}(x^{(t)}, u^{(t)})$ given the state and control vectors $(x^{(\tau)}, u^{(\tau)})_{\tau\le t-1}$. These completely determine $x^{(t)}$, given by equation (3), and the subproblem becomes

$$\min_{u^{(t)}\in\Delta^u(\alpha,F)} \quad J^{(t)}(x^{(t)}, u^{(t)}). \qquad (5)$$

The controller anticipates the move of the selfish players and myopically optimizes the objective on the next iteration. This is a convex optimization problem, since $J^{(t)}$ is convex by assumption. It can be solved using mirror descent. The greedy solution can then be summarized as follows:

Algorithm 2 Greedy method

1: $x^{(0)}$ is given.
2: **for** each time step $0 \le t \le T$ **do**
3: $\quad u^{(t)} = \arg\min_{u\in\Delta^u(\alpha,F)} J^{(t)}(x^{(t)} + u)$
4: $\quad$ Update $x^{(t+1)}$ according to equation (3)
5: **return** $u = (u^{(t)})_{1\le t\le T}$

B. Complexity analysis

The greedy method solves T convex optimization problems on the product of simplices $\Delta^u(\alpha, F)$, where each problem is followed by an update of the selfish distribution. According to the previous section, the number of iterations of mirror descent on $\mathcal{P}_k$ grows as $\mathcal{O}(\ln|\mathcal{P}_k|)$, where each iteration requires computing the gradient of the objective $J^{(t)}$, then updating the distribution, which has a linear cost $\mathcal{O}(|\mathcal{P}_k|)$. Thus the total complexity of each problem is $\mathcal{O}(|\mathcal{P}_k|\log|\mathcal{P}_k|)$, and the total complexity of the greedy method is $\mathcal{O}(T\sum_{k=1}^K |\mathcal{P}_k|\log|\mathcal{P}_k|)$.

V. The adjoint method

In this section, we propose to use the adjoint method to find a local minimum of the non-convex problem (2). First, we can reformulate problem (2) as follows:

$$\begin{aligned} \underset{u^{(t)}\in\Delta^u(\alpha,F)}{\text{minimize}} \quad & J(x,u) \\ \text{subject to} \quad & H(x,u) = 0 \end{aligned} \qquad (6)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

where we used a function H defined on $\mathbb{R}^{T|\mathcal{P}|} \times \mathbb{R}^{T|\mathcal{P}|}$ with values in $\mathbb{R}^{T|\mathcal{P}|}$ to encode the constraints on the selfish flow distribution x, given by the initial distribution x^{init} and the Hedge update equations (3).

The adjoint method is a general local search method for solving optimal control problems under non-linear constraints, of the form given in problem (6). It is derived using the stationarity conditions of Pontryagin's maximum principle. For an introduction to the adjoint method in optimal control, see for example [24], [25] and references therein. A complete exposition of the adjoint method is beyond the scope of this article, but we give below a formal derivation and intuitive interpretation of the adjoint system equations.

Since the control u entirely determines the selfish flow distributions x, let us assume (for illustration purposes) that the state x can be written as $x = X(u)$ for some differentiable function $X : \times_{t=1}^{T} \Delta^u(\alpha, F) \to \times_{t=1}^{T} \Delta^x(\alpha, F)$. The optimal control problem would then be equivalent to minimizing the function $J(X(u), u)$ over the feasible set $\times_{t=0}^{T} \Delta^u(\alpha, F)$, and we can use the mirror descent algorithm to solve this problem, since the constraint set is a product of simplices. To apply mirror descent, we need to compute, at each iteration, the gradient of the function $u \mapsto J(X(u), u)$, which we denote $\nabla_u J(x, u)$. Using the chain rule, we have the following expression of the gradient

$$\nabla_u J(x,u) = \frac{\partial J}{\partial x}(x,u)\nabla_u X(u) + \frac{\partial J}{\partial u}(x,u), \tag{7}$$

where the Jacobian term $\nabla_u X(u)$, which represents the dependence of the state x on the input u, can be expensive to compute. The adjoint method provides a different approach to computing the gradient (7) without explicitly computing the state Jacobian: Since $H(X(u), u) = 0$, we have, taking derivatives,

$$\frac{\partial H}{\partial x}(x,u)\nabla_u X(u) + \frac{\partial H}{\partial u}(x,u) = 0. \tag{8}$$

Therefore if we let λ be a solution to the system

$$\left[\frac{\partial H}{\partial x}(x,u)\right]^T \lambda = -\left[\frac{\partial J}{\partial x}(x,u)\right]^T, \tag{9}$$

we have

$$\begin{aligned}\lambda^T \frac{\partial H}{\partial u}(x,u) &= -\lambda^T \frac{\partial H}{\partial x}(x,u)\nabla_u X(u) \\ &= \frac{\partial J}{\partial x}(x,u)\nabla_u X(u),\end{aligned}$$

where we used (8) in the first equality and (9) in the second. Plugging this expression in (7), we obtain the following expression of the gradient

$$\nabla_u J(x,u) = \lambda^T \frac{\partial H}{\partial u}(x,u) + \frac{\partial J}{\partial u}(x,u). \tag{10}$$

Our method iteratively solves the adjoint equations (9) to compute the gradient (10), and performs one mirror descent step in the direction of this gradient. This summarized in Algorithm 3, where we use the superscript $[i]$ to denote step i in the algorithm, not to be confused with superscript (t), which denotes time t (corresponding to one term of the objective function).

Algorithm 3 Mirror descent using the adjoint method

1: Initialize $i = 0$, $u^{[0]} \in \times_{t=1}^{T} \Delta^u(\alpha, F)$, and $x^{[0]}$ by solving $H(x^{[0]}, u^{[0]}) = 0$.
2: **while** stopping criterion not satisfied **do**
3: Solve the adjoint system

$$\left[\frac{\partial H}{\partial x}(x^{[i]}, u^{[i]})\right]^T \lambda^{[i]} = -\left[\frac{\partial J}{\partial x^{[i]}}(x^{[i]}, u^{[i]})\right]^T$$

Compute the gradient

$$g^{[i]} = \nabla_u J(x^{[i]}, u^{(i)}) = \lambda^{[i]T} \frac{\partial H}{\partial u}(x^{[i]}, u^{[i]}) + \frac{\partial J}{\partial u}(x^{[i]}, u^{[i]}).$$

4: Perform one mirror descent step: for each k and each $p \in \mathcal{P}_k$,

$$u_p^{k\,[i+1]} \propto u_p^{k\,[i]} \exp(-\beta_i g_p^{k\,[i]}).$$

5: Update $x^{[i+1]}$ by solving $H(x^{[i+1]}, u^{[i+1]}) = 0$.
6: Update $i \leftarrow i + 1$
7: **return** Control solution $u^{[i_{\text{best}}]}$.

We run the method from multiple random initial points $u^{(0)}$ and keep the best local minimum.

A. Derivation of the adjoint system equations for the Hedge dynamics

In this section, we explicitly derive, for the Hedge learning dynamics, the adjoint system equations (9). This derivation is of general interest since it applies to any optimal control problem that is subject to the Hedge dynamics, not only to our optimal routing problem. Under Hedge dynamics, the constraints H can be written as follows: $\forall k$ and $\forall p \in \mathcal{P}_k$,

$$H_p^{k\,(0)}(x,u) = x_p^{k\,(0)} - x_p^{k\,(init)},$$

and for $t \geq 1$,

$$\begin{aligned}H_p^{k\,(t)}(x,u) = x_p^{k\,(t)} &- \\ (1-\alpha_k)F_k &\frac{x_p^{k\,(t-1)} e^{-\eta_{t-1}\ell_p^k(x^{(t-1)}+u^{(t-1)})}}{\sum_{q\in\mathcal{P}_k} x_q^{k\,(t-1)} e^{-\eta_{t-1}\ell_q^k(x^{(t-1)}+u^{(t-1)})}}.\end{aligned} \tag{13}$$

To simplify the derivation, let

$$\begin{aligned}\ell_{p,q}'^{(t-1)} &= \frac{\partial \ell_p(x^{(t-1)}+u^{(t-1)})}{\partial x_q^{(t-1)}} = \frac{\partial \ell_p(x^{(t-1)}+u^{(t-1)})}{\partial u_q^{(t-1)}} \\ &= \sum_{e\in E} \tilde{M}_{e,p}\tilde{M}_{e,q}c_e' \left([\tilde{M}(x^{(t-1)}+u^{(t-1)})]_e\right), \\ w_p^{k\,(t-1)} &= \exp(-\eta_{t-1}\ell_p^k(x^{(t-1)}+u^{(t-1)})), \\ W^{k\,(t-1)} &= \sum_{q\in\mathcal{P}_k} x_q^{k\,(t-1)} \exp(-\eta_{t-1}\ell_q(x^{(t-1)}+u^{(t-1)})).\end{aligned}$$

First, let us calculate $\frac{\partial H_p^{k\,(t)}}{\partial x_q^{k\,'(s)}}(x,u)$. Since $H^{(t)}$ only depends on $x^{(t)}$, $x^{(t-1)}$ and $u^{(t-1)}$, we have $\frac{\partial H_p^{k\,(t)}}{\partial x_q^{k\,(s)}}(x,u) = 0$ except

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

$$\frac{\partial H_p^{k(t)}}{\partial x_q^{k'(t-1)}} = -(1-\alpha_k)F_k w_p^{k(t-1)}\left[x_p^{k(t-1)}\eta_{t-1}\left(\frac{\ell'^{(t-1)}_{p,q}}{W^{k(t-1)}} - \frac{\sum_{r\in\mathcal{P}_k} x_r^{k(t-1)} w_r^{k(t-1)} \ell'^{(t-1)}_{r,q}}{(W^{k(t-1)})^2}\right) + \delta_k^{k'} x_p^{k(t-1)} \frac{w_q^{k'(t-1)}}{(W^{k(t-1)})^2} - \delta_{p,q}\frac{1}{W^{k(t-1)}}\right] \tag{11}$$

$$\frac{\partial H_p^{k(t)}}{\partial u_q^{k(t-1)}} = -(1-\alpha_k)F_k w_p^{k(t-1)} x_p^{k(t-1)}\eta_{t-1}\left(\frac{\ell'^{(t-1)}_{p,q}}{W^{k(t-1)}} - \frac{\sum_{r\in\mathcal{P}_k} x_r^{k(t-1)} w_r^{k(t-1)} \ell'^{(t-1)}_{r,q}}{(W^{k(t-1)})^2}\right) \tag{12}$$

for $s \in \{t-1, t\}$. Then we have $\frac{\partial H_p^{k(t)}}{\partial x_q^{k'(t)}} = \delta_k^{k'}\delta_p^q$, where we use the Kronecker notation $\delta_k^{k'} = 1$ if $k = k'$ and 0 otherwise. The expression for $\frac{\partial H_p^{k(t)}}{\partial x_q^{k'(t-1)}}$ is given in equation (11) at the top of the page. Similarly, $\frac{\partial H_p^{k(t)}}{\partial u_q^{k'(s)}}$ is zero except when $s = t-1$, and the expression for $\frac{\partial H_p^{k(t)}}{\partial u_q^{k(t-1)}}$ is given in equation (12).

B. Complexity analysis

The method performs a mirror descent on the product of simplices $\times_{t=1}^{T}\Delta^u(\alpha, F)$, and the number of iterations on each simplex $\mathcal{P}_k$ grows as $\mathcal{O}(\log|\mathcal{P}_k|)$, as in the greedy method. However, now each gradient evaluation requires solving the adjoint system (9) then using the expression (10) of the full gradient. One gradient evaluation thus requires calculating the partial derivatives $\frac{\partial J}{\partial u}(x,u), \frac{\partial J}{\partial x}(x,u) \in \mathbb{R}^{T|\mathcal{P}|}$, and $\frac{\partial H}{\partial x}(x,u), \frac{\partial H}{\partial u}(x,u) \in \mathbb{R}^{T|\mathcal{P}|\times T|\mathcal{P}|}$, then solving the adjoint system (9) for $\lambda \in \mathbb{R}^{T|\mathcal{P}|}$. If we further assume that the cost function $J^{(t)}$ at time-step t only depends on $x^{(t)}$ and $u^{(t)}$, then the matrix $\frac{\partial H}{\partial x}(x,u)$ is banded lower-triangular, and contains $\mathcal{O}(T|\mathcal{P}|^2)$ non-zero terms. Therefore solving the adjoint system can be done in $\mathcal{O}(T|\mathcal{P}|^2)$ using Gaussian elimination, as discussed for example in [26] Appendix C-2. Therefore, the total computational complexity of the adjoint method scales as $\mathcal{O}\left(T|\mathcal{P}|^2\sum_{k=1}^{K}\ln|\mathcal{P}_k|\right)$. It is linear in T, similarly to the greedy method, but scales quadratically (up to logarithmic factors) in the total number of paths $|\mathcal{P}|$.

VI. Performance on a simple example

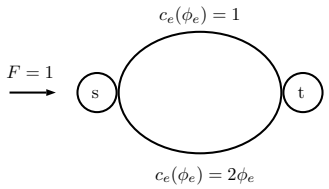
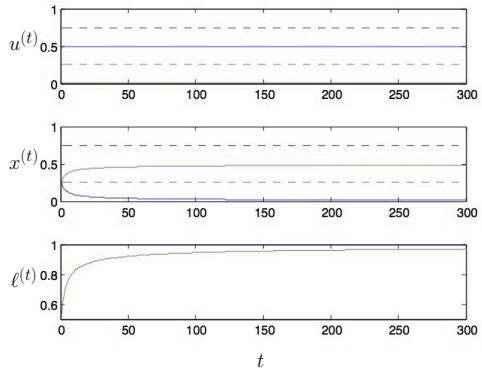


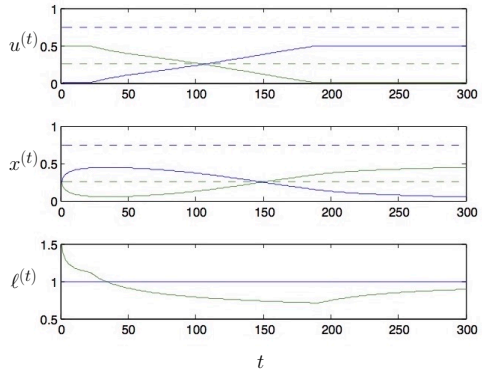
Fig. 1: Pigou network used for the numerical experiment.

To illustrate the qualitative difference between the greedy and the adjoint solutions, we consider a simple example known as the Pigou network, given in Figure 1.

It is given by a single population of players with total mass $F_1 = 1$, and with two paths connecting the origin to the destination, each consisting of a single edge. The congestion function on the top edge is constant, $c_1(\phi_1) = 1$, and the congestion function on the second edge is linear, $c_2(\phi_2) = 2\phi_2$. This routing game has a unique Nash equilibrium, given by $x^{\text{Nash}} = (\frac{1}{2}, \frac{1}{2})$ (under this equilibrium, both edges have the same loss). The total delay of the network (as defined in Example 1) is $x_1 + 2x_2^2 = x_1 + 2(1-x_1)^2$, which is minimal at $x^{(\text{social})} = (\frac{1}{4}, \frac{3}{4})$.



(a) Greedy solution.

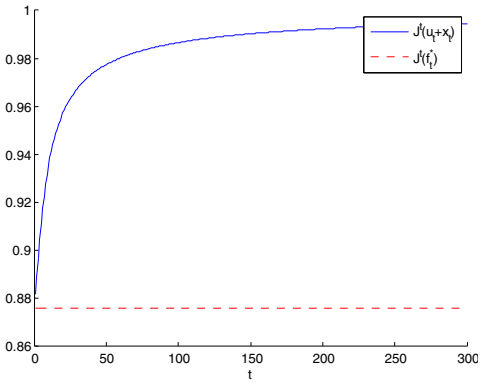


(b) Adjoint solution.

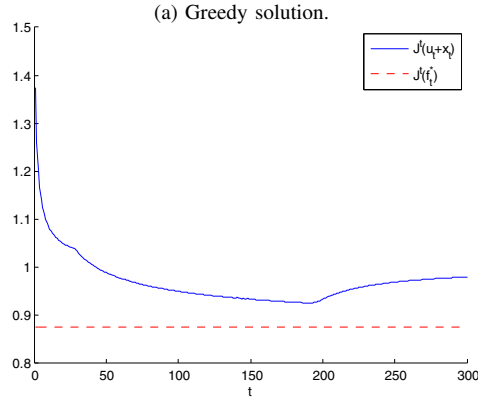
Fig. 2: Control solution on the Pigou network. Controlled mass $u^{(t)}$ (top), selfish mass $x^{(t)}$ (middle) and corresponding path losses $\ell^{(t)}$ (bottom). The green lines correspond to the top path, and the blue lines to the bottom path. The dashed lines show the social optimum $x^{\text{social}} = (\frac{1}{4}, \frac{3}{4})$.

The particularity of this example is that whenever $\alpha \leq \frac{1}{2}$, no Stackelberg strategy can improve the total cost, since any allocation of the controlled flow will induce the same total

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems



(a) Greedy solution.



(b) Adjoint solution.

Fig. 3: Social cost $J^{(t)}$ over time induced by adjoint solution and the greedy solution. The dashed line shows the social optimal allocation.

flow distribution, equal to $(\frac{1}{2}, \frac{1}{2})$. So for the one-shot game, the equilibrium of the system cannot be improved. However, one may take advantage of the selfish learning dynamics to reduce the cost on a finite horizon.

Consider the total delay on the network, $J^{(t)}(x,u) = \sum_{p\in\mathcal{P}_1}(x_p^1 + u_p^1)\ell_p^1(x_p^1 + u_p^1)$, and fix $\alpha = \frac{1}{2}$. We assume that the selfish population obeys the Hedge dynamics, starting from the uniform distribution $x^{(0)}$. Without control, the selfish flow distribution is stationary, $x^{(t)} = (\frac{1}{2}, \frac{1}{2})$ for all t (note that this corresponds to the Nash equilibrium). We simulate the greedy method and the adjoint method with $T = 300$, $\alpha = \frac{1}{2}$, and learning rates $\eta_t = \frac{1}{\sqrt{t}}$. The greedy and the adjoint solutions are illustrated in Figure 2 and 3, and are quite different qualitatively: The greedy solution assigns all the controlled flow to the upper path at all times, $u^{(t)}_{\text{greedy}} = (\frac{1}{2}, 0)$ for all $t \in \{1, \dots, T\}$, since this is the best myopic decision at any time. The adjoint solution, however, first allocates flow to the lower path, which decrease the selfish flow on that path. Then, the controlled flow is moved to the upper path, which results in decreasing the cost on the lower path. This strategy achieves a better social cost ($J^{\text{greedy}} = 291.7$; $J^{\text{adjoint}} = 283$; the no-control cost ($\alpha = 0$) is $J^{\text{selfish}} = 300$; and the social optimal cost ($\alpha = 1$) is $J^* = 262.5$). The per-time-step costs $J^{(t)}$ for both solutions are given in Figure 3, where we can observe that the adjoint solution sacrifices the cost on the first few time-steps for a lower cost on later time steps. In particular, this example illustrates the limitations of the greedy approach, since, by definition, it does not anticipate the dynamics of the selfish population over several time steps.

VII. Numerical experiments on synthetic data

In this section, we compare the performance and running time of each method on parallel networks of increasing-size. Parallel networks can be used to model job scheduling problems (for example in [5]). This simple topology also allows us to evaluate how each method scales, and provides some insights on the qualitative behavior of the solution.

A. Experimental setting

We consider parallel networks with $|\mathcal{P}|$ paths. We fix the following parameters of the models: time horizon $T = 20$, total flow $F = 0.5$, control fraction $\alpha = 0.1$, learning rates of the selfish players $\eta_t = \frac{1}{\sqrt{t}}$. We run the different methods on networks of increasing size, $|\mathcal{P}| \in \{2, 5, 10, 20, 40, 60\}$. For a given network size $|\mathcal{P}|$, we run $N = 100$ instances of the simluation, each time with a different set of congestion functions $\{c_e\}_{e\in\mathcal{P}}$, and output the quantities given in Table I below. The congestion functions are taken to be linear, of the form $c_e(\phi_e) = a_e\phi_e + b_e$, where a_e and b_e are drawn randomly in $[0, 1]$. The social cost function J is taken to be the total delay function (4).

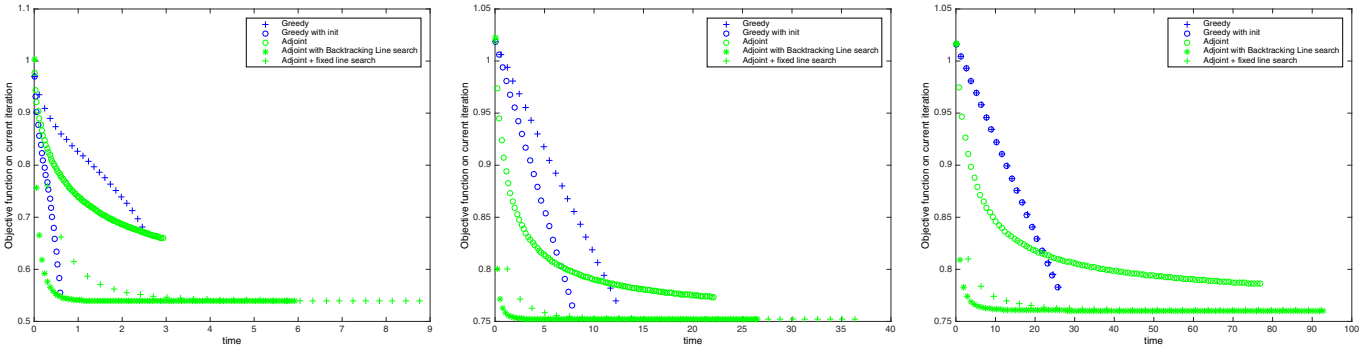
J^{selfish}	Value of the objective under selfish flows ($\alpha = 0$).
J^*	Value of the objective under optimal flows ($\alpha = 1$).
$J^{m[i]}$	Value of the objective function at the i^{th} iteration of method m.
$P^{m[i]}$	Normalized objective value, $\frac{J^{m[i]} - J^*}{J^{\text{selfish}} - J^*}$.
$\tau^{m[i]}$	The running time of the i^{th} iteration of method m.

TABLE I: Output of one simulation.

We then report the average value of the scaled objective on Fig. 4, and the average running time in Fig. 5. We scale the objective value by $J^{\text{selfish}} - J^*$, so that the values on different network instances are comparable.

For the greedy method (Section IV), each iteration i corresponds to the minimization of one term of the objective function: Recall that the objective can be decomposed into $J(x,u) = \sum_{t=1}^{T} J^{(t)}(x^{(t)}, u^{(t)})$, and that the greedy algorithm minimizes one term of the objective at a time. Thus at iteration i, the greedy algorithm computes a partial greedy solution $(u^{(1)}, \dots, u^{(i)})$ (in particular, $(u^{(i+1)}, \dots, u^{(T)})$ are not computed yet). In order to report the value of the objective at step i, we simply evaluate the objective on the vector $(u^{(1)}, \dots, u^{(i)}, u_0, \dots, u_0)$, where $u_0 \in [\Delta^u(\alpha, F)]$ is the uniform distribution. For the adjoint method, each iteration i corresponds to one step of mirror descent on $[\Delta^u(\alpha, F)]^T$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

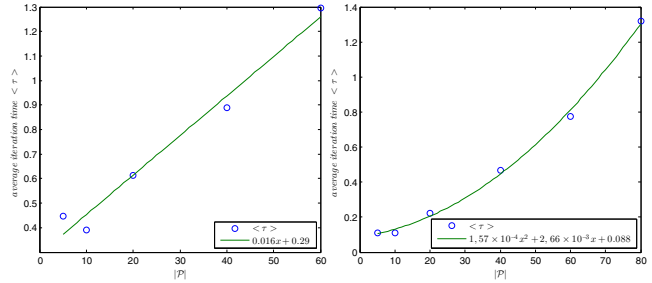


(a) $|\mathcal{P}| = 2$ (b) $|\mathcal{P}| = 20$ (c) $|\mathcal{P}| = 60$

Fig. 4: Average performance of each method on a random instances of a parallel networks of size $|\mathcal{P}|$.

B. Results

We can see on Fig. 4 that, except in the special case of 2-path networks, a control of $\alpha = .1$ of the total flow allows a $\sim 25\%$ reduction in the social cost, compared to the selfish dynamics. The greedy and adjoint methods achieve a comparable value of the objective function. The adjoint method with backtracking line search achieves the best performance, and requires fewer iterations to converge.



(a) Greedy Algorithm (b) Adjoint

Fig. 5: Average time of one iteration as a function of the size $|\mathcal{P}|$ of the network and corresponding polynomial fit. The results confirm our complexity analysis: the greedy method scales linearly in $|\mathcal{P}|$, and the adjoint method scales quadratically.

VIII. Numerical experiment on the Los Angeles highway network

A. Los Angeles highway network

In this section, we consider a model of the Los Angeles highway road network, used in [28], and illustrated in Fig. 6[1]. The network topology is obtained from OpenStreetMap data, by keeping highways that contain five lanes or more. We consider $K = 42$ source-destination pairs (illustrated in Fig. 6b), for the following destinations: Hollywood (node 5), Santa Monica (node 20) and Central L.A. (node 22). In order to limit the problem size, we also restrict the set of paths available to each souce-destination pair, by removing any path that is empty under both social optimum and Nash equilibrium.

The congestion functions are those estimated by the Bureau of Public Roads for a network in quasi-static equilibrium [28]. More precisely, the congestion function is assumed to be of the form $c_e(\phi_e) = d_e D(\phi_e/m_e)$, where $D(x) = 1 + 0.15x^4$, ϕ_e is the edge flow, d_e a free-flow delay, and m_e is the capacity on edge e. The simulations are run with a time horizon $T = 20$, learning rates $\eta_t = \frac{1}{\sqrt{t}}$, with values of $\alpha \in \{.1, .3, .5, .7, .9, 1\}$. The results are given in Fig. 7 and 8, and discussed bellow.

B. Effect of increasing control

First, we observe that increasing the control parameter α results in a decrease in the total delay, and for higher values of α, the value of the objective is very close to that of the social optimum. Although intuitive, this cannot be guaranteed in general, since the problem is non-convex for all α strictly between 0 and 1, so the problem may converge to a worse local minimum for a lower value of α.

C. Numerical Results for $\alpha = 0.1$

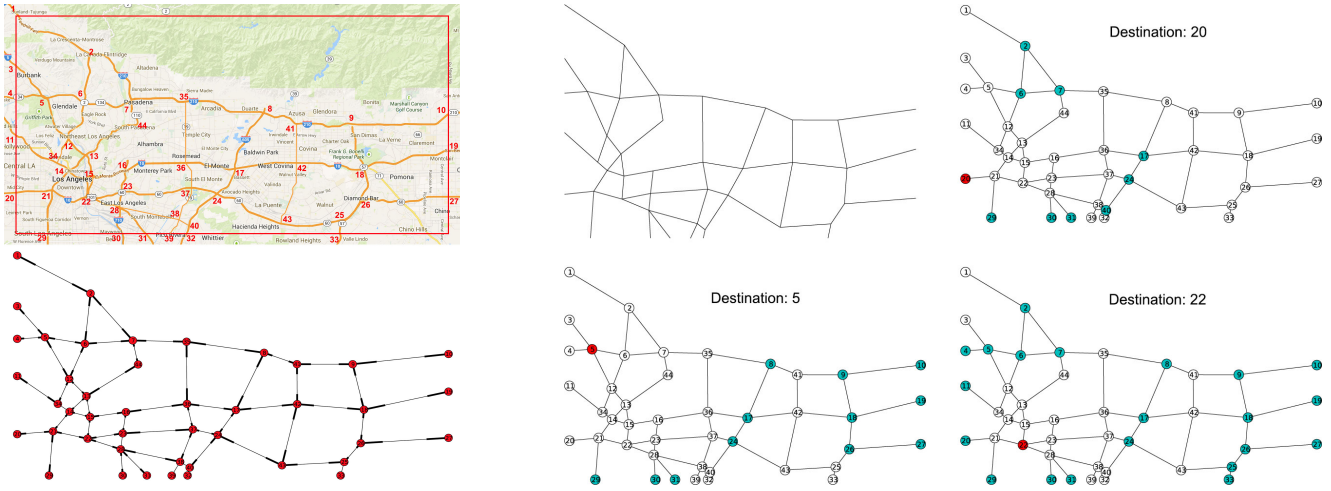
We now have a more detailed look at the performance of each method with a fixed $\alpha = 0.1$. The values of the objective function for each method are reported in the following table, and the average delay per vehicle per day is reported as a function of iteration number in Fig. 8.

J^{social}	J^{greedy}	J^{adjoint}	$J^{\text{adjoint\_ls}}$	J^{selfish}
25.4	29.0	28.7	32.0	36.7

We observe that most of the methods do not guarantee a decrease in the value of the objective from one iteration to the next, except the adjoint method with line search, which, by definition, searches for a step size which guarantees a descent (by Armijo's rule). Nevertheless, the adjoint method without line search performs best, and converges to a local

[1]The model and the data used to generate it are taken from Thai et al. [28]. The code is available at https://github.com/jeromethai/traffic-estimation-wardrop

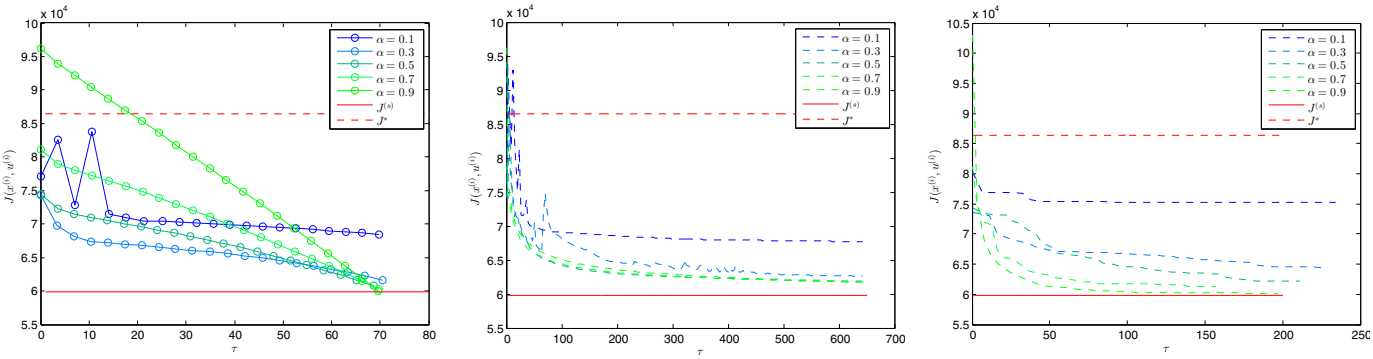
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems



(a) L.A. highway network and its graph model.

(b) Selected sources (blue) and destinations (red) for the Los Angeles highway network

Fig. 6: Los Angeles highway network.



(a) Greedy method

(b) Adjoint method

(c) Adjoint method with backtracking line search

Fig. 7: Results for the different methods, with increasing α. The red solid and dotted lines represent, respectively, the social optimum ($\alpha = 1$) and the selfish response ($\alpha = 0$).

minimum with lower objective value than that with line search. This may be a result of line search being too conservative: Requiring the Armijo rule to be satisfied at each iteration may prevent the method from exploring the search space. The greedy method performs surprisingly well, and is within 3% of the (normalized) objective value of the adjoint method. Finally, it is worth observing that even when controlling a fraction of the population as small as $\alpha = 0.1$, the improvement in the social cost function can be significant (70% reduction in the distance to social optimum).

IX. Conclusion

We studied a problem of repeated routing under selfish response, in which the selfish players follow online learning dynamics given by the Hedge algorithm. Such dynamics offer a realistic model of behavior, and are asymptotically consistent with Nash equilibria. Subject to these dynamics, the problem of optimal routing is non-convex, and cannot be solved exactly in general. We proposed two methods to approach this non-convex problem, and analyzed their computational complexity and their performance on numerical examples: A greedy method and a local search method based on the adjoint system. In particular, we derived the adjoint system equations associated to the Hedge dynamics.

Our numerical experiments shed light on the tradeoffs and the empirical performance of each method: The adjoint method has the best performance, but its complexity is quadratic. The greedy method, while limited due to its myopic nature, is simple to implement and performs well empirically.

While our application was specific to the routing game, the derivation of the Hedge adjoint equations is generic and can be applied to any optimal control problem where a selfish population is assumed to follow Hedge dynamics. Our experiments indicate that the adjoint method works well in practice, and encourages further investigation into the numerical performance on other applications.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

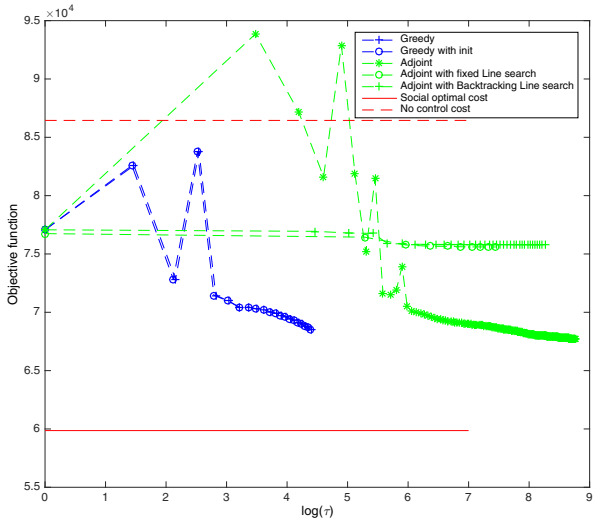


Fig. 8: Average delay (minutes) per vehicle and per day, with $\alpha = .1$, as a function of time.

REFERENCES

[1] T. Roughgarden and É. Tardos, "How bad is selfish routing?" *Journal of the ACM (JACM)*, vol. 49, no. 2, pp. 236–259, 2002.

[2] T. Roughgarden, "Routing games," in *Algorithmic game theory*. Cambridge University Press, 2007, ch. 18, pp. 461–486.

[3] A. Ozdaglar and R. Srikant, "Incentives and pricing in communication networks," *Algorithmic Game Theory*, pp. 571–591, 2007.

[4] Y. A. Korilis, A. A. Lazar, and A. Orda, "Capacity allocation under noncooperative routing," *IEEE Transactions on Automatic Control*, vol. 42, pp. 309–325, 1997.

[5] T. Roughgarden, "Stackelberg scheduling strategies," *SIAM Journal on Computing*, vol. 33, no. 2, pp. 332–350, 2004.

[6] D. Pisarski and C. Canudas-de Wit, "Optimal balancing of road traffic density distributions for the cell transmission model," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 2012, pp. 6969–6974.

[7] A. Blum, E. Even-Dar, and K. Ligett, "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*, ser. PODC '06. New York, NY, USA: ACM, 2006, pp. 45–52.

[8] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press, 2006.

[9] J. Hannan, "Approximation to Bayes risk in repeated plays," *Contributions to the Theory of Games*, vol. 3, pp. 97–139, 1957.

[10] J. R. Marden, "Regret based dynamics: Convergence in weakly acyclic games," in *In Proceedings of the 2007 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2007.

[11] S. Hart and A. Mas-Colell, *Simple Adaptive Strategies: From Regret-matching to Uncoupled Dynamics*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2012.

[12] J. Lions, *Optimal control of systems governed by partial differential equations*, ser. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1971.

[13] M. B. Giles and N. A. Pierce, "An introduction to the adjoint approach to design," *Flow, Turbulence and Combustion*, vol. 65, no. 3-4, pp. 393–415, 2000.

[14] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications." *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[15] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Oper. Res. Lett.*, vol. 31, no. 3, pp. 167–175, May 2003.

[16] J. G. Wardrop, "Some theoretical aspects of road traffic research." in *ICE Proceedings: Engineering Divisions*, vol. 1, no. 3. Thomas Telford, 1952, pp. 325–362.

[17] R. Kleinberg, G. Piliouras, and E. Tardos, "Multiplicative updates outperform generic no-regret learning in congestion games," in *Proceedings of the 41st annual ACM symposium on Theory of computing*. ACM, 2009, pp. 533–542.

[18] W. Krichene, B. Drighès, and A. Bayen, "On the convergence of no-regret learning in selfish routing," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. JMLR Workshop and Conference Proceedings, 2014, pp. 163–171.

[19] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.

[20] J. R. Marden and J. S. Shamma, "Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation," *Games and Economic Behavior*, vol. 75, no. 2, pp. 788 – 808, 2012.

[21] L. E. Blume, "The statistical mechanics of strategic interaction," *Games and Economic Behavior*, vol. 5, no. 3, pp. 387 – 424, 1993.

[22] K. Lam, W. Krichene, and A. Bayen, "On learning how players learn: Estimation of learning dynamics in the routing game," in *7th International Conference on Cyber-Physical Systems (ICCPS)*, 2016.

[23] A. S. Nemirovsky and D. B. Yudin, *Problem complexity and method efficiency in optimization*, ser. Wiley-Interscience series in discrete mathematics. Wiley, 1983.

[24] L. C. Evans, "An introduction to mathematical optimal control theory."

[25] E. Polak, *Optimization: Algorithms and Consistent Approximations*. New York, NY, USA: Springer-Verlag New York, Inc., 1997.

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*, C. U. Press, Ed. Cambridge University Press, 2010, vol. 25.

[27] S. Dempe, *Foundations of Bilevel Programming*. Springer US, 2002.

[28] J. Thai, R. Hariss, and A. Bayen, "A multi-convex approach to latency inference and control in traffic equilibria from sparse data," in *American Control Conference (ACC), 2015*, July 2015, pp. 689–695.

APPENDIX A
THE MIRROR DESCENT ALGORITHM

Consider the problem of minimizing a convex function f on a convex feasible set $\mathcal{X}$. The mirror descent algorithm minimizes, at each iteration, a local approximation of the function f, as detailed in Algorithm 4, where β_i is a predefined sequence of decreasing step sizes. We can also use a line search method for choosing the step sizes, as discussed in Appendix B. The mirror descent method is guaranteed to converge to the set of minimizers $\arg\min_{\mu\in\mathcal{X}} f(\mu)$ for example when β_i decreases to 0 and $\sum_{i=1}^{\infty}\beta_i = \infty$, see for example [15].

Algorithm 4 Mirror descent algorithm

for $t \in \mathbb{N}$ **do**

Compute a subgradient $\ell^{[i]} \in \partial f(\mu^{[i]})$

$$\mu^{[i+1]} = \arg\min_{\mu\in\mathcal{X}} \left[\left\langle \ell^{[i]}, \mu - \mu^{[i]} \right\rangle + \frac{1}{\beta_i} D_\psi(\mu, \mu^{[i]}) \right] \quad (14)$$

Here, D_ψ is the Bregman divergence induced by a strongly convex function ψ, and given by $D_\psi(\mu,\nu) = \psi(\mu) - \psi(\nu) - \langle \nabla\psi(\nu), \mu - \nu\rangle$. The choice of the Bregman divergence can be adapted to the geometry of the feasible set $\mathcal{X}$. In particular, if $\mathcal{X}$ is a simplex, $\mathcal{X} = \Delta^n$, and we choose ψ to be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2619910, IEEE Transactions on Control of Network Systems

the negative entropy, $\psi(\mu) = H(\mu) := \sum_i \mu_i \ln \mu_i$, then the corresponding Bregman divergence is the KL divergence, $D_{KL}(\mu, \nu) := \sum_i \mu_i \ln \frac{\mu_i}{\nu_i}$, and the solution of the mirror descent update in Algorithm 4 is exactly the Hedge update given in equation (1), see [15]. Furthermore, when $\mathcal{X}$ is a cartesian product of simplices, $\mathcal{X} = \Delta^{\mathcal{P}_1} \times \cdots \times \Delta^{\mathcal{P}_K}$, then we can take ψ to be the sum of negative entropies, $\psi(\mu) = \sum_{k=1}^K H(\mu^k)$, in which case the Bregman divergence is the sum of KL divergences, $D_\psi(\mu, \nu) = \sum_{k=1}^K D_{KL}(\mu^k, \nu^k)$, and the minimization problem (14) decomposes into K minimization problems, each on a simplex $\mathcal{P}_k$.

APPENDIX B
LINE SEARCH IN MIRROR DESCENT

In the standard implementation of Hedge, we run the algorithm with a predefined sequence of step sizes, which results in a relatively slow convergence to the desired precision. Line search is a method for adaptively choosing the step size β_i at each iteration. We implement the backtracking line search method, commonly used in projected gradient descent and analyzed for example in Chapter 9 in [26]. We adapt this method to Hedge. This is summarized in Algorithm 5.

Algorithm 5 Backtracking line search for Hedge with Armijo coefficient $a \in (0, \frac{1}{2})$, backtracking rate $b \in (0, 1)$

1: Input: previous iterate $u^{[i]}, x^{[i]}$ and previous gradient vector $g^{[i]} = \nabla_u J(x^{[i]}, u^{[i]})$.
2: Initialize $\beta_i = \beta_i^{\text{init}}$
3: **while** $J(x^{[i+1]}, u^{[i+1]}) \geq J(x^{[i]}, u^{[i]}) + a \left\langle g^{[i]}, u^{[i+1]} - u^{[i]} \right\rangle$ **do**
4: Solve for $u^{[i+1]}$

$$u_p^{k\,[i+1]} \propto u_p^{k\,[i]} \exp(-\beta_i g_p^{k\,[i]})$$

5: Update $x^{[i+1]}$ by solving $H(x^{[i+1]}, u^{[i+1]}) = 0$
6: Update $\beta_{i+1} \leftarrow b\beta_i$

The Armijo condition in step 4 above can be justified by writing the first order Taylor approximation of J around the previous iterate: $J(X(u), u) = J(x^{[i]}, u^{[i]}) + \left\langle g^{[i]}, u - u^{[i]} \right\rangle + \mathcal{O}(\|u - u^{[i]}\|^2)$, and evaluated at $u^{[i+1]}$,

$$J(x^{[i+1]}, u^{[i+1]}) \leq J(x^{[i]}, u^{[i]}) + \left\langle g^{[i]}, u^{[i+1]} - u^{[i]} \right\rangle + C\|u^{[i+1]} - u^{[i]}\|^2, \quad (15)$$

for some positive constant C. Now since $u^{[i+1]}$ is, by definition of the Hedge update, the minimizer of $\left\langle g^{[i]}, u - u^{[i]} \right\rangle + \frac{1}{\beta_{i+1}} D_\psi(u, u^{[i]})$, we have

$$\begin{aligned} &\left\langle g^{[i]}, u^{[i+1]} - u^{[i]} \right\rangle + \frac{1}{\beta_{i+1}} D_{KL}(u^{[i+1]}, u^{[i]}) \\ &\leq \left\langle g^{[i]}, u^{[i]} - u^{[i]} \right\rangle + \frac{1}{\beta_{i+1}} D_{KL}(u^{[i]}, u^{[i]}) = 0. \end{aligned}$$

Thus $\left\langle g^{[i]}, u^{[i+1]} - u^{[i]} \right\rangle$ is negative, and

$$\begin{aligned} D_\psi(u^{[i+1]}, u^{[i]}) &\leq -\beta_{i+1} \left\langle g^{[i]}, u^{[i+1]} - u^{[i]} \right\rangle \\ &\leq \beta_{i+1} \|g^{[i]}\|_\infty \|u^{[i+1]} - u^{[i]}\|_1. \end{aligned}$$

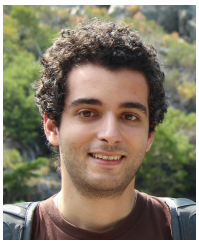
Finally, by Pinsker's inequality, we have $D_{KL}(u^{[i+1]}, u^{[i]}) \geq \frac{1}{2}\|u^{[i+1]} - u^{[i]}\|_1^2$, and it follows that $\frac{1}{2}\|u^{[i+1]} - u^{[i]}\|_1^2 \leq \beta_{i+1}\|g^{[i]}\|_\infty \|u^{[i+1]} - u^{[i]}\|_1$. Simplifying, we have

$$\|u^{[i+1]} - u^{[i]}\|_1 \leq 2\beta_{i+1}\|g^{[i]}\|_\infty,$$

which proves that as $\beta_{i+1} \to 0$, $\|u^{[i+1]} - u^{[i]}\| \to 0$, and the Taylor bound (15) is eventually dominated by the Armijo bound

$$\begin{aligned} J(x^{[i]}, u^{[i]}) + \left\langle g^{[i]}, u^{[i+1]} - u^{[i]} \right\rangle + C\|u^{[i+1]} - u^{[i]}\|^2 \\ < J(x^{[i]}, u^{[i]}) + a \left\langle g^{[i]}, u^{[i+1]} - u^{[i]} \right\rangle, \end{aligned}$$

for $\|u^{[i+1]} - u^{[i]}\|$ small enough, which justifies using the Armijo rule for Hedge.



Walid Krichene is a Ph.D. candidate in the department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. His research focuses on the theory of convex optimization, and the study of learning dynamics in distributed systems, with human and computer agents. He received the M.S. in Applied Mathematics from the Ecole des Mines Paristech and the M.A. in Mathematics from the University of California, Berkeley. His previous research work includes collaboration with the CDC department at Caltech, the CAOR group at the Ecole des Mines, and a researcher position at Criteo labs. Walid has received multiple awards including the Valedictorian Prize for the Tunisian Baccalaureate, a Bronze medal at the Pan African Mathematics Olympiads, two Outstanding Graduate Student Instructor awards from UC Berkeley, and the Leon Chua Award for outstanding achievement in nonlinear science.

Milena Suarez is an applied researcher in the Economic Studies Department at INSEE (French national statistics institute). Her research focuses on public policy evaluation and econometrics. She received an Engineer-Economist degree from the Ecole Nationale de la Statistique et de l'Administration Economique, a Master of Advanced Studies in Mathematics from Cambridge University where she received a College Examination Prize from Trinity College. She is also a graduate from Ecole Polytechnique. She was a visiting student in Professor Bayen's lab at the University of California, Berkeley in 2014.

Alexandre Bayen is the Liao-Cho Professor of Engineering at UC Berkeley, in Electrical Engineering and Computer Sciences, and Civil and Environmental Engineering. He is the Director of the Institute of Transportation Studies, and a Faculty Scientist at the Lawrence Berkeley National Laboratory. He received the Engineering Degree from the Ecole Polytechnique, France, 1998, the M.S. and Ph.D. from Stanford University in 1999 and 2003 respectively. He was a Visiting Researcher at NASA Ames Research Center from 2000 to 2003. Between January 2004 and December 2004, he worked as the Research Director of the Autonomous Navigation Laboratory at the Laboratoire de Recherches Balistiques et Aerodynamiques, (Ministere de la Defense, Vernon, France), where he holds the rank of Major. He authored two books and over 200 peer reviewed publications. He is the recipient of the Best of ITS Award, the TRANNY Award from the California Transportation Foundation, the Ballhaus Award from Stanford University, the CAREER Award from the National Science Foundation, the Presidential Early Career Award for Scientists and Engineers (PECASE) from the White House, the Okawa Research Grant Award, the Ruberti Prize from the IEEE, and the Walter Huber Prize from the ASCE.