Speed Traps and Ticket Dodging: Multiagent Decision-Making on Transportation Networks

Ernesto Nunes, Julio Godoy, and Maria Gini
Dept of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
{enunes, godoy, gini}@cs.umn.edu

ABSTRACT

We model a transportation network where agents of different types operate with conflicting objectives: drivers want to drive at high speeds to reach their destination faster, while police agents want to prevent unlawful speeding. Police units have to efficiently allocate their limited resources to monitor roads and catch speeders, who try to avoid being caught. Assuming that police and drivers make strategic choices, the problem can be modeled as a game-theoretic problem. We describe the models and algorithms we developed and validate them on different maps.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Game Theory, Adversarial Reinforcement Learning, Experience-Weighted Attraction

1. INTRODUCTION

The problem of guarding an area or a network against adversaries when opponents can see guards is well studied. However, the problem becomes hard when the locations of the adversaries are not known, the number of guards is limited, and everyone adapts dynamically to changes in the environment. Catching drivers who speed is an example of such a problem.

We model the problem of speed control as a general-sum repeated stochastic game played by police units and drivers in a transportation network. Police units are in charge of preventing drivers from driving above the speed limit. Drivers speed up to reach their destination faster and try to avoid getting caught by the police. Our choice of modeling the problem as a simultaneous game is motivated by the fact that it is natural to assume drivers will not know police mixed strategies prior to making their speeding decision (lack of observability) and drivers can simultaneously speed along different roads (different targets). This makes the use of Stackelberg leadership models not appropriate [13].

The game does not have pure strategy equilibrium points, and takes a long time to converge to a mixed-equilibrium [12]. Hence, we are interested in the dynamics during the game. We show empirically that police and driver agents can exploit each other when suboptimal choices are made.

The algorithm we present is inspired by Experience-Weighted Attraction (EWA) [4]. In our model, the algorithm, which combines dynamic programming and EWA learning principles, is used by drivers to learn the paths where the chance of being caught when speeding is the smallest. A simpler version of the algorithm without the dynamic programming part is used by police units to decide where to position themselves to increase the probability of catching drivers when they speed. We compare experimentally the performance of our algorithm against Opponent Q-learning [15] on two city maps, a synthetic map and a real city map, using different numbers of agents.

The main contribution of this paper is the modeling of the problem and our learning algorithm. Our study of the dynamics that ensue from interactions among drivers and police units can be used to produce guidelines for police deployment on transportation networks. The game we present is a complex game with many heterogeneous agents, and can be used to test other multiagent learning algorithms.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

2. RELATED WORK

Opportunities and challenges for the use of agents in traffic control are outlined in [3]. Most work in this area focuses on adaptive control of traffic lights (e.g., [1]) and intersections [6], or on modeling individual drivers' behaviors [9]. Kim et al. [12] investigate if increased penalties decrease illegal speeding. They model opponent strategies as population mixed strategies, where police behavior is influenced by the proportion of drivers receiving a ticket, while driver behavior is influenced by police presence. Hence, while individual agents follow pure strategies, population aggregate statistics are used to introduce stochastic choices. Our work takes a step further and investigates how agents learn to play the game, while also learning the topology of the underlying transportation network. To the best of our knowledge, multiagent opponent learning in transportation networks to catch unlawful speeders remains a sparsely studied topic.

There is a large body of work on security deployment and patrolling. Patrolling or monitoring units are typically placed in strategic locations to either respond to adversarial activity or prevent it. Game-theoretic algorithms are used in [8] and [7] to patrol a fence or an area. The problem is modeled as a Stackelberg game when attackers can observe the strategy defenders play before acting [8, 11]. We have chosen to model the problem as a simultaneous game because we do not assume drivers know police strategies prior to making decisions on speeding.

The actions of police units and drivers can be framed as a K-armed bandit problem, where the objective is to choose which gambling machine to play to maximize the payoff [2]. In our case, random variables are the roads upon which agents make choices. Our driver agents could plan their paths as in [14], in which loop-free stochastic shortest-paths are computed using multiarmed bandit solutions.

Our work benefits from the diverse literature on opponent learning (e.g., [15, 16]) and EWA learning [4, 10]. The design of our agents is inspired by the principles presented in Camerer [4] and Uther [15].

3. BACKGROUND

3.1 Definitions, Assumptions and Notation

The *environment* is a network of roads represented by a weighted graph $G = \{V, E, W\}$. A node $v \in V$ represents an intersection between two roads and an edge $e \in E$ represents a road. We consider both directed and undirected graphs, to account for one- and two-way roads. Each edge has a weight $w \in W$, which represents the travel time through that road segment at the speed limit. With some abuse of notation we will use the terms node and state interchangeably.

The *game* is defined as $\mathcal{G} = (\mathcal{N}, \{\mathcal{S}_i\}_{i=1}^n, \{\mathcal{U}_i\}_{i=1}^n)$, where $\mathcal{N} = \{1, ..., n\}$ is the set of agents, $\mathcal{C} \subset \mathcal{N}$ is the set of police agents and $\mathcal{D} \subset \mathcal{N}$ is the set of driver agents. $\{\mathcal{S}_i\}_{i=1}^n$ is the set of pure strategies and $\{\mathcal{U}_i\}_{i=1}^n$ is the payoff function for each agent.

For police agent $j \in \mathcal{C}$ a pure strategy $s_\ell \in \mathcal{S}_j$ has the format $s_\ell =$ (to-node, action). Where action is from the set $\mathcal{A}_j = \{\text{enforce, not enforce}\}$. Police agents either move to another node or stay at the current node and choose an action from $\mathcal{A}_j$. Police agents can choose to enforce a ticket only if a driver is speeding. For instance, at node v_1 a police agent can choose $s_1 = (v_1, \text{enforce})$ or $s_2 = (v_2, \text{not enforce})$. When choosing s_1 the police agent does not move and catches In s_2 the police moves to node v_2 and sees a speedy driver, it decides not to issue a ticket according to some probability. Placing probabilities over $\mathcal{A}_j$ allows us to model the uncertainty real police experience when giving a ticket.

Likewise, a pure strategy $s_q \in \mathcal{S}_k$ for driver agent for $k \in \mathcal{D}$ has the format $s_q =$ (to-node, action), where action is from the set $\mathcal{B}_k = (\text{speed} \leq L, L < \text{speed} \leq L + 10, \text{speed} > L+10)$, where L is the speed limit. A pure strategy $s_q =$ (to-node, action) for a driver agent differs from s_ℓ because driver agents always have to move. Hence, $s_1 = (v_3, \text{speed}), s_2 = (v_2, \text{not speed})$ are examples of strategies for driver agent who is at node v_1.

When playing in a complete graph with m nodes, the set of pure strategies $\mathcal{S}_i$ is as large as $|\mathcal{A}|m(m-1)$ for police agents and $|\mathcal{B}|m(m-1)$ for drivers agents. and $|\mathcal{A}|m(m-1)$ for police agents.

A *mixed strategy* for agent $i \in \mathcal{N}$ is a probability distribution over the pure strategies $\mathcal{S}_i$. Our proposed approach uses EWA to compute these probability distributions.

A *joint play* for agent i represents a simultaneous move by i and its opponents. A move is comprised of the pure strategy agent i plays and a vector of the strategies played by its opponents.

A *neighborhood* of a node is the set of adjacent nodes. There might be zero or more police agents in a neighborhood, and a police agent might be at a node that is on the path of several drivers.

Each *game iteration* starts with all driver agents at their initial locations, and ends when all drivers have reached their destinations. The game is repeated to enable agents to learn from previous games and better predict opponent strategies.

At the end of each game, for each node and each agent, a summary of the outcomes is stored in a *History*, which is used by the agents to make decisions in successive games. To reduce the amount of information to make a decision, agents are only allowed to access the history in their neighborhood.

After a joint play, all agents receive a payoff according to the payoff matrix (Table 1). The payoff depends on the speeding option and the length of the road. We introduce the payoff function in Section 4. We assume the following *preferences*: (1) If no police agent is predicted to be at the next node, a driver agent prefers to *speed above 10 mph* ($f_2 > e_2 > d_2$). (2) If police is predicted at the next node, drivers prefer not to speed ($a_2 > b_2$). (3) Police agents almost always enforce a ticket if drivers *speed above 10 mph* ($c_1 > b_1 > a_1$) but are less likely do so if drivers *speed below 10 mph*. They get zero payoff when drivers do not speed. The probabilities of enforcing tickets are discussed later in Section 5.

	Driver		
Police	speed$\leq L$	$L<$speed$\leq L+10$	speed$>L+10$
E	a_1, a_2	b_1, b_2	c_1, b_2
N	d_1, d_2	e_1, e_2	e_1, f_2

Figure 1: Payoff Matrix. *E=Enforce, N= Not enforce*

Further, we assume that agents do not communicate among themselves and that driver agents speed solely to shorten their travel time.

3.2 Learning Algorithms

The learning principles of our algorithm are inspired by EWA [4]. Police agents learn by combining soft-max action selection with the updating rules of EWA, while driver agents combine EWA and soft-max action selection with a Dijkstra-like algorithm to pick the best road and speeding decision. We use Opponent Q-learning [15] as a benchmark algorithm.

Experience-Weighted Attraction: Experience-Weighted Attraction [4] combines two learning models, belief and choice reinforcement. Three data parameters are kept: the agent's experience, which is measured as the discounted observations of past opponent plays; the agent's attractions to strategies; and the probabilities of the agent playing its strategies. There are also three weight parameters: the forgetting parameter (ϕ) which takes values in [0,1] and is used as a discount factor for observations. If ϕ=1 the agent remembers the opponents past plays, if ϕ=0 it forgets them. The attention parameter (δ), with values in [0,1], represents the attention an agent pays to foregone payoffs. $\delta = 0$ means that the agent reinforces the chosen strategies with a weight of 1, and the not chosen strategies with a weight in [0,1] [4]. The response sensitivity parameter (λ) allows the agent to either pick a random response (if $\lambda = 0$) or to best respond (if $\lambda = \infty$).

Agents update their experiences and attractions according to Eqs. (1) and (2).

$$N(t) = \phi(1-\kappa)N(t-1) + 1, t \geq 1 \tag{1}$$

$$A_i(s_\ell, t) = \frac{\phi N(t-1)A(s_\ell, \boldsymbol{O}_{-i}, t-1)}{N(t)} + \frac{[\delta + (1-\delta)I(s_\ell)U_i(s_\ell, \boldsymbol{O}_{-i})}{N(t)} \tag{2}$$

Agent i computes its attraction to pure strategy $s_\ell \in \mathcal{S}_i$ as a combination of its prior experience, prior attraction, and the payoff collected from a joint play $((s_\ell, \boldsymbol{O}_{-i}))$ [10], where $\boldsymbol{O}_{-i}$ is the set of pure strategies chosen by agent i's opponents. I is an indicator function that returns 1 if $\mathbf{s}_\ell$ is the strategy chosen by agent i in the previous iteration of the game, and 0 otherwise. The chosen strategies are reinforced with their full payoff and the not chosen strategies with a fraction (δ) of their payoffs. Agents are allowed to explore according to the exploration parameter (κ). The probability of agent i picking strategy s_ℓ is computed via a logit function (Eq. 3).

$$P_i(s_\ell, t) = \frac{e^{-[\lambda A_i(s_\ell,t)]}}{\sum_{\forall s_x \in \boldsymbol{O}_{-i}} e^{-[\lambda A_i(s_x,t)]}}, \forall s_x \neq s_\ell \tag{3}$$

Opponent Q-learning: Opponent Q-learning [15] extends the standard Q-learning algorithm (Eqs. 4–6) by learning optimal policies in the presence of opponents [16].

$$Q(v,a) = R(v,a) + \gamma \sum_{v'} P(v,a,v')V(v') \tag{4}$$

$$V(v) = \max_a(Q(s_\ell)) \tag{5}$$

$$Q(v,a) = U_i(s_\ell) + \gamma V(v') \tag{6}$$

$P(v,a,v')$ is the probability for agent i of transitioning from its current node v to its next node v' with action a. Since in our case all actions for drivers are deterministic, Eq. 4 is equivalent to Eq. 6, where the reward can now be linked to both start and destination nodes, by performing the corresponding action. Agents form beliefs about how opponents play using the observed frequency with which the opponent played an action to estimate the probability of the opponent choosing that action in the future. The probability is factored into the calculation of the expected value of the action in that node (Eq. 8). The update of each agent current beliefs (Eq. 7) uses two parameters, α and γ, which are the learning rate and the discount factor on future actions, respectively [15].

$$Q(v,a) = \alpha Q(v,a) + (1-\alpha)(U_i(v,a) + \gamma V(v')) \tag{7}$$

$$E[Q(v,a)] = \sum_{\forall s_x \in \boldsymbol{O}_{-i}} P(s_x|v)Q(s_\ell|s_x) \tag{8}$$

Eq. (8) computes the expected value of an action by summing across the expected values of the joint play of the agent and its opponents. In order to select the next node, the agent computes the action with the highest expected value $\text{argmax}_{s_\ell} E[Q(v,a|s_x)]$, accounting for $P(s_x|v)$, where $P(s_x|v)$ is the probability that the opponent plays the strategy s_x from node v. The value of this action is the value $V(v')$ for that node. This method of choosing the action given the probability distribution of previous actions is the same as arriving at a best action using fictitious play [16].

4. DESIGN OF LEARNING AGENTS

4.1 Environment Representation

The environment model is an augmented graph with three types of edges: and a *non-speeding edge*, a *speeding up to 10* edge, and a *speeding above 10* edge. Edges represent possible driver actions and are added between any pair of nodes that are connected in the original graph (see Fig. 2).

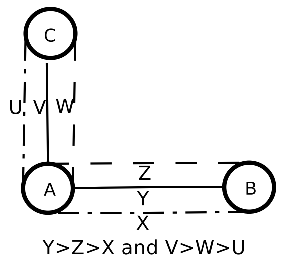


Figure 2: Graph-based road model. Edges represent travel time.

Each edge is weighted by the travel time to traverse it and produces a different payoff or penalty, proportional to its weight and outcome of the play. Let $W(s_\ell)$ be a function that returns the weight of the edge connecting the agent's current node v and node v' of strategy s_ℓ, according to the speeding decision of the strategy. Let $H(v)$ be the function that returns the weight of the heaviest non-speeding edge from v. The payoff driver agent d gets when moving from v to v' using s_ℓ is defined as:

$$U_d(s_\ell) = \begin{cases} 1 - \frac{W(s_\ell)}{H(v)} & \text{if no ticket is received} \\ 0 & \text{if not speeding} \\ -1 & \text{if ticket received} \end{cases} \tag{9}$$

The payoff for police agent c stationed at node v is as follows:

$$U_c(s_\ell) = \begin{cases} 1 \text{ if driver speeds, is caught, and gets ticket} \\ 0 \text{ if driver speeds, is caught, but gets no ticket} \\ 0 \text{ if driver does not speed} \\ -1 \text{ if driver speeds, but there is no police} \end{cases} \tag{10}$$

4.2 EWA-Inspired Agent Models

Dynamic-EWA Driver Agent Design: At the start of the game each driver agent begins with initial beliefs on its own strategies. In each iteration the algorithm (Algorithm 1) follows two steps: (1) it computes an optimistic path using Dijkstra's shortest path algorithm and simultaneously use EWA to choose the actions; (2) it updates the agents' beliefs using EWA learning principles. The Dijkstra shortest path algorithm computes paths that minimize travel, assuming that drivers will always speed on the edges, hence the optimistic assumption. However, the speeding decision is made using EWA probabilities (Eq. 3). In the worst case, the agent may choose not to speed on all the edges in the path provided by Dijkstra's algorithm, resulting in a slower path. Using Dijkstra's shortest path algorithm frees agents from having to learn which paths to follow, instead, agents focus on which decisions to make on the paths.

Another advantage is that drivers get loop-free paths at a relatively low computational cost. The algorithm keeps a priority queue (q) to reduce computation time for the minimum weight edge. The distance ($dist(s)$) for a node v is the distance from the initial node ($start$) to v. If v' is a neighbor of v, $dist(v')$ is updated to be $dist(v)+\text{weight}(v,v')$, if this is less than the prior value for $dist(v')$. Nodes already visited are not visited again to avoid loops and re-computations. The computed path can be retrieved from a data structure that saves the predecessor of each node in the path ($prev$). The EWA model is updated according to driver and police strategies on nodes in the path. Payoff for each edge is computed according to Eq. (9). Payoffs and historical data are used to compute the new values for Eqs. (1)–(3).

Algorithm 1 Dynamic-EWA Driver

```
 1: Initialize attraction, experience, and beliefs
 2: for each iteration on games $t$ do
 3:     $visited = \emptyset$
 4:     for $v \in V$ do
 5:         $dist(v) = \infty$
 6:     $current = start$; $dist(current) = 0$
 7:     $q = q \cup (current, dist[current])$
 8:     while $q$ not empty do
 9:         $current = argmin(q)$
10:         $q = q - target$
11:         $visited = visited \cup target$
12:         for $n \in neighbors(current)$ do
13:             if $n \notin visited$ then
14:                 $dst = dist(current) + \text{weight}(current, n)$
15:                 if $dist(n) > dst$ then
16:                     $prev(n) = current$
17:                     $dist(n) = dst$
18:                     $q = q \cup (n, dst)$
19:                     Choose an action with P from Eq. (3)
20:     Compute the path using prev
21:     Update attraction, experience and probabilities
22:     Play according to the path and chosen actions
```

EWA-Inspired Police Agent Design: A police agent follows the steps described in Algorithm 2. Upon catching a driver, depending on whether the driver is *10 mph* above the speed limit or not, police agents choose to enforce a ticket or not according to a probability distribution (see Section 5). If a police agent moves, its movement is constrained to a neighboring node. The constraint helps reduce the strategy space over which the agent has to sample from the Cartesian product of the number of nodes and number of available actions, to the Cartesian product of the number of neighbor nodes and number of available actions. This reduction can be significant for large graphs.

Analysis of the Algorithms: The dynamic-EWA algorithm incurs computational costs during path selection, update of the learning model, and selection of the best-response strategy. During path selection, the dynamic-EWA algorithm requires a $O(N^2)$ worst-case running time, where N is the number of nodes. When running on completely connected graphs, the algorithm inspects N nodes and $N-1$ neighbors. Driver agents compute the best-response strategy by evaluating the space of joint plays between each driver and the entire police population. While this space

Algorithm 2 Police EWA-based strategy selection

```
1: Initialize attraction, experience, and beliefs
2: Choose initial strategy $(start, start, a_i)$ with P from
   Eq. (3)
3: for each iteration $t$ do
4:     for each $v' \in \{v \cup neighbors(v)\}$ do
5:         for each $s_\ell = (v', a_i)$ do
6:             Observe payoffs given $\boldsymbol{O}_{-i}$
7:             Update history and the EWA model
8:     Choose $s_\ell$ P from Eq. (3)
```

could be large, our algorithm makes two simplifying assumptions in order to reduce complexity: (1) driver agents only need to consider strategies of police agents in the neighborhood of the driver. (2) Driver agents are independent in relation to other driver agents, hence they plan paths independently of each other. The first assumption allows agents to prune the strategy space to consider. The size of the reduced strategy space is define by the Cartesian product of the agent's strategies by the opponent's strategies, both constrained to the neighborhood of the current agent location. For police agents using EWA, the learning algorithm is cheaper than that of drivers, because the main cost comes from the evaluation of the space of joint strategies.

The algorithms do not converge to pure strategy equilibrium points because such points do not exist in the game [12]. Driver agents drive above the speed limit if police agents do not enforce tickets. The increased number of speeding drivers leads police agents to give tickets, which in turn causes driver agents not to speed. Hence, either agent type can increase its payoff by changing its strategy unilaterally. Intuitively, the game would have an equilibrium when the probabilities of enforcing a ticket makes the driver agent choose to speed or not with equal likelihood. Proving that the algorithms converge to such equilibrium distributions is left for future research.

4.3 Opponent Q-learning Models

Opponent Q-learning Driver Agent Design: The Q-values associated with each node in the graph represent the discounted payoff of speeding or not speeding in each outgoing edge. We store only one Q-value for speeding and one for not speeding to reduce the number of Q-values stored. Drivers use thresholds to decide whether to speed or not: driver agents choose to *speed above 10 mph* if the Q-value for speeding is larger than the Q-value for not speeding. They choose to *speed up to 10 mph* if the Q-value for speeding is greater than half of the Q-value for not speeding (for the same action and next node pair). Else, if the Q-value for speeding is less than half of the value for not speeding, then the agent decides *not to speed.* For any two neighboring nodes, the two related Q-values are updated using the Bellman equation (7) [15] modified as follows: when drivers choose to speed, the discounted value of the action is multiplied by one minus the frequency with which tickets were issued at a node.

In this model, drivers do not plan paths to destination beforehand. Instead, drivers make local node-level decisions to learn the path to their goals. Additionally, drivers rely on their history to deduce police presence when making a speeding decision.

Opponent Q-learning Police Agent Design: Police agents select their next node according to the following criteria:

- Spatial criteria – select only neighbor nodes each time.
- Expected profit – select another node only if the expectation of issuing more tickets is higher than the actual rate of tickets issued in the present node.
- Presence or absence of police – select a next node that is not already occupied by another police agent.

In each iteration, each police agent checks these three conditions for each neighboring node and moves only if there is a node with a higher expected payoff.

5. EXPERIMENTS

We designed a set of experiments to test the effectiveness of the algorithms.

Graphs: We chose two graphs with similar structures but different sizes and number of agents: a 4×4 graph, which we call the Grid graph, and a portion of a US city, which we call the Downtown graph.

The Grid graph (Fig. 3) has a rich set of paths. Twelve drivers start on the first two rows of the graph, and have different destinations in the bottom row. Six police agents are placed in the bottom rows. The graph has 16 nodes and 72 edges, and contains loops. We use this graph to analyze drivers behaviors when there are multiple paths to destination and how police adapt to them.

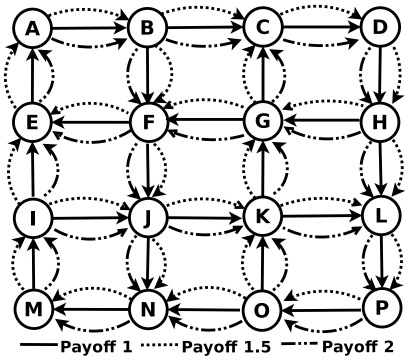


Figure 3: Grid graph, with symmetrical weights on the edges

The Downtown graph (Fig. 4) contains 279 nodes, with edges that reflect the real traffic directions in the city. We use six drivers and three police agents. The start and destination nodes for the drivers are landmark places in the city. This map was chosen to study how the size of the decision space affects the prediction ability of the agents.

In Fig. 4 we show the paths for two drivers, $D1$ and $D2$. The blue and green circles indicate their start and destination nodes. The two police agents, $P1$ and $P2$, are represented by red circles. Because $P2$ is in the path of $D2$, $D2$ will eventually get ticketed if it decides to speed on the road segment that leads to the location of agent $P2$. For both graphs we ran 30,000 iterations per experiment.

In addition to EWA-based police algorithm, we tested three other algorithms for police agents:

- Adaptive: police agents analyze the profit in their current node and in neighboring nodes, based on the number of observed speeding drivers. They decide to move if a neighboring node seems more profitable in terms of the number of tickets that they may issue.
- Static: the police agents do not change node.
- Random: the police agents move to random nodes. The only constraint that restricts their movement is the presence of another police agent at the randomly chosen node.

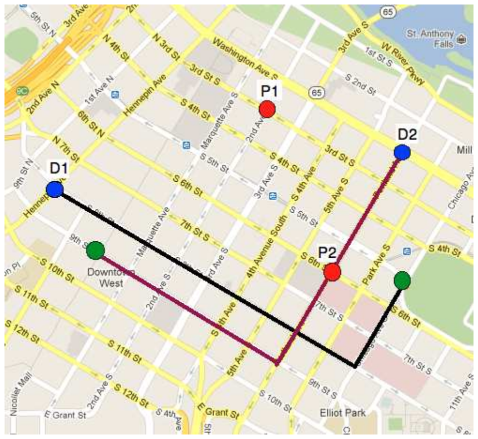
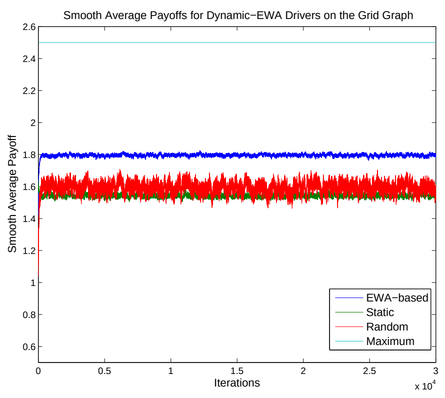


Figure 4: Map of a section of the downtown of a US city, with paths of two drivers and two police agents.

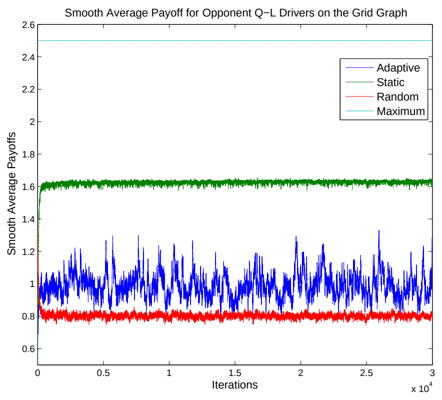
Algorithm Parameters: We set the probability of police agents to issue tickets to drivers that *speed above 10 mph* to be 0.9, and drivers that *speed below 10 mph* to be 0.3. These values reflect the assumption that driving at higher speeds increase the chance of getting a ticket. For the EWA-inspired algorithms, we found experimentally that $\lambda = 0.45$ for the driver and $\lambda = 0.65$ for the police agents lead to better performance. The attraction decay rate (ϕ) for a strategy was automatically set by subtracting from one the ratio between the number of times the opponent played a strategy so far divided by the number of iterations so far. Strategies the opponents play less often will then have higher experience values according to Eq. (1). The exploration parameter $\kappa = 0.65$ allows agents to explore moderately. The weight on forgone payoffs (δ) was also set automatically as follows: if the potential payoff of playing a strategy is greater than the payoff of playing any other strategy, then $\delta = 1$, else $\delta = \frac{\phi}{constant}$, with $constant = 2$.

Experiments with the Opponent Q-learning drivers show that setting α to 0.2 gives more weight to newly computed values, allowing for faster convergence of the Q-values. γ weights the best neighboring Q-value so that it prefers better actions in earlier steps of the game than later. Experiments showed that $\gamma = 0.6$ yielded better results than other values. *VisitDiscount* is used to prevent driver agents from going into a cycle when the presence of a police prevents them from reaching their destination. After a road is visited, we multiply its Q-values by *VisitDiscount*=0.5; this makes future visits to that road less likely. To enable driver agents to share the same road without influencing each other, the discounted value is computed for each agent separately.

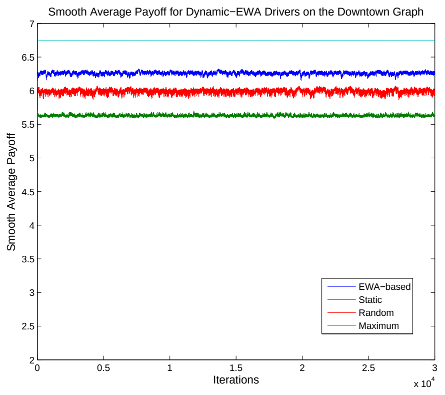
Metrics for the Experiments: We use two metrics in our experiments. The payoff driver agents accumulate during the iterations of the game and the regret driver agents experience. Regret is measured as the difference between the best possible payoff and the payoff the agent receives from



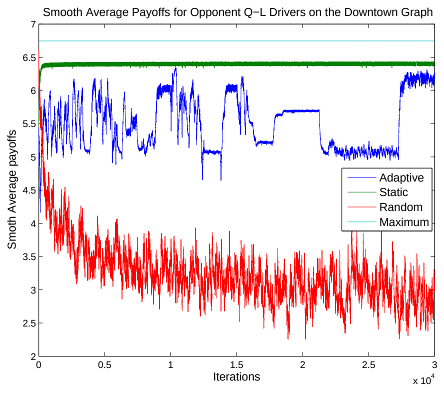
(a) Dynamic-EWA on the Grid Graph



(b) Opponent Q-L on the Grid Graph



(c) Dynamic-EWA on the Downtown Graph



(d) Opponent Q-L on the Downtown Graph

Figure 5: Payoffs for Dynamic-EWA and Opponent Q-learning drivers.

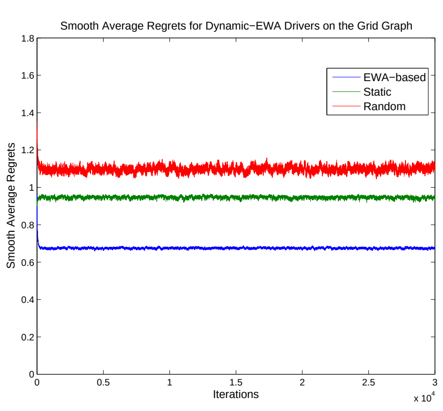
the joint play. The highest regret has a value of 1 and occurs when a driver is caught speeding. Drivers that do not speed do not experience regret. The regret for a driver agent that speeds below *10 mph* and is not caught is computed as the difference between the payoff for speeding above *10 mph* and the payoff for speeding below *10mph*. We report smoothed payoffs and regrets. This helps to better see the trends in the average values of these quantities. In addition, payoffs and regret are not tied to any specific metric, they solely represent agent preferences.

Data Generation. The data we used are synthetic. The speeding and non-speeding travel times were created according to our discretion for the Grid graph, but we used real distances for the Downtown map and computed the travel time by multiplying the edge length by one time unit (assumes agents travel one meter per time unit). To compute the travel time when drivers speed, we randomly generated the travel times for speeding below and above 10mph. Future research will involve using real traffic and ticket data.

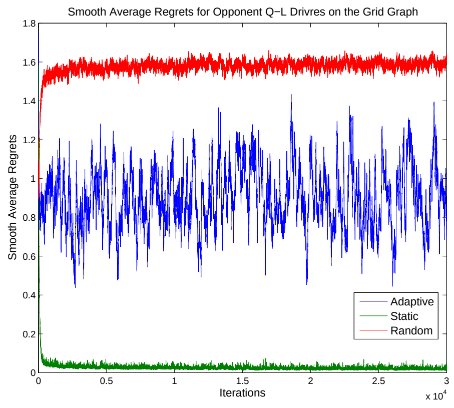
6. RESULTS

Next, we report results of experiments performed on the Grid and the Downtown graphs.

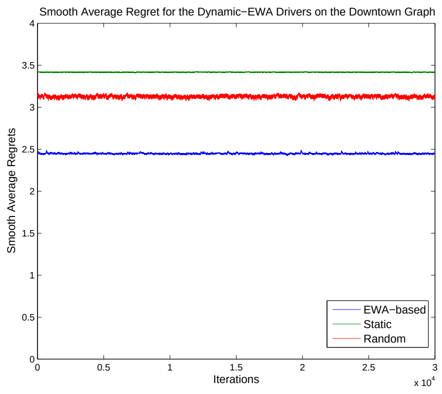
Grid graph. Results in Fig. 5(a) and Fig. 5(b) illustrate an example in which adaptive driver agents perform poorly when playing against police agents that play randomly. The average payoff that dynamic-EWA and Opponent Q-learning driver agents collect is lower when playing against police that play random strategies than when playing against the EWA-based and adaptive police agents, respectively. Police playing randomly choose each location on the map with equal probability. Perfect randomization produces a near uniform distribution of the frequency with which police visit nodes on the graph. This makes it harder for driver agents to effectively predict strategies police agents will play in the future. The average payoff for dynamic-EWA driver agents converges to an average payoff of roughly 1.6 (after smoothing the payoffs) when playing against a random police allocation. The payoffs for dynamic-EWA drivers are roughly twice as large as the average payoffs of Opponent Q-learning driver agents (0.8). The difference in average payoffs indicates that dynamic-EWA drivers adapt better against random police than Opponent Q-learning drivers. The dynamic-EWA algorithm takes advantage of combining the history of joint plays and the changes in collected rewards to compute probabilities for choosing the action. This enables dynamic-EWA drivers to better randomize against opponents. Opponent Q-learning agents garner their highest payoffs when playing against static agents (1.6 units of payoff). To the contrary, dynamic-EWA drivers perform poorly against static drivers. The poor performance can be



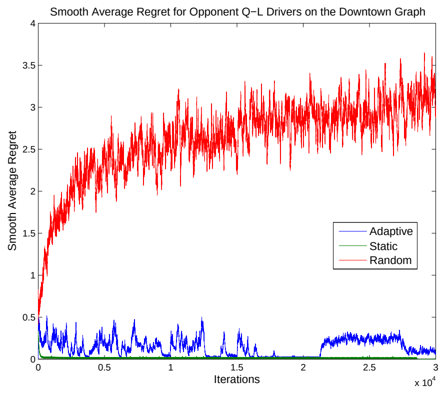
(a) Dynamic-EWA on the Grid Graph



(b) Opponent Q-L on the Grid Graph



(c) Dynamic-EWA on the Downtown Graph



(d) Opponent Q-L on the Downtown Graph

Figure 6: Regrets for Dynamic-EWA and Opponent Q-learning drivers

explained by observing that when using soft-max selection players do not choose the best-response strategy with probability 1. Instead, they choose the most attractive strategy with higher probability [5]. Hence, the probabilistic best-response may allow driver agents to choose to speed towards a node, even if police were present at the node in the recent past. This happens when the probability of playing the best choice is close in value to the probabilities of playing other strategies.

Opponent Q-learning drivers experience highest regret when playing against police that play randomly (Fig. 6(b)). This is consistent with the reported payoff results. Driver agents become conservative in the presence of random police and choose not to speed on roads in which they received tickets. Contrarily, drivers' regret is minimized when playing against static police. This illustrates a case in which Opponent Q-learning drivers exploit police suboptimal decisions by speeding where police are not present and not speeding in locations where they received tickets in the past. The average regret Opponent Q-learning drivers experience is cyclic when playing against adaptive police, evidencing adaptive behavior. Dynamic-EWA drivers experience the highest regret when playing against police agents that play randomly. This is consistent with the payoffs agents receive. Both dynamic-EWA and Opponent Q-learning drivers experience similar regrets (with the mean of the average regrets between 0.6 to 0.8) when playing against the EWA-based police and the adaptive police, respectively.

The large police to driver ratio (1 : 2) in this experimental setup leads driver agents to obtain low payoffs. This example illustrates the fact that larger police presence leads drivers to loose more from suboptimal decisions.

Next, we consider a larger graph with a layout similar to the Grid graph but with fewer police units. Results show a narrower gap between drivers' payoffs and the maximum attainable average payoff.

Downtown Graph: Driver agents that use Dynamic-EWA and Opponent Q-learning attain similar average payoffs when playing against EWA police and adaptive police (Fig. 5(c) and Fig. 5(d)). The payoffs are close to the maximum attainable. The limited movement of both types of adaptive police agents combined with the number of uncovered paths driver agents can take are the main reasons for the success of these driver agents. In this experiment, two of the three police agents start at a location that intersects driver shortest paths in at most one edge. However, these police agents cover less area, which allows driver agents to speed without punishment on these uncovered edges. The same is not true when playing against police agents that play randomly. These agents can see the whole map, and they are allowed to fly over to other locations. Dynamic-EWA drivers perform better than Opponent Q-learning drivers when playing against police agents that play randomly.

Dynamic-EWA drivers perform better for two main reasons: first, dynamic-EWA drivers only travel on the shortest paths returned by the Dijkstra's algorithm. Thus, agents explore fewer paths than Opponent Q-learning drivers. Exploring fewer paths makes it less likely for police agents choosing randomly to occupy nodes on the driver's path with more frequency than nodes outside the path. Hence, dynamic-EWA agents take advantage of the absence of police agents to speed and collect higher rewards.

Second, dynamic-EWA driver agents are risk-takers. These agents might choose to speed, even in nodes where they previously received a ticket, provided that the node has a higher attraction than the recently played nodes. The regret results for the dynamic-EWA (Fig. 6(c)) and for the Opponent Q-learning driver (Fig. 6(d)) agents confirm that dynamic-EWA algorithms do worse against static police, while Opponent Q-learning drivers are able to learn to play against this type of police agents. It also confirms that both agents struggle against police playing random strategies.

7. CONCLUSIONS AND FUTURE WORK

We used repeated stochastic games to model the interaction between police and driver agents in a transportation network. We proposed the dynamic-EWA algorithm that enables drivers to adapt to police agents. Experimental results indicate that the proposed approach performs well against police agents playing EWA and random strategies. Our results support that the deployment of police force in strategic locations discourages drivers from speeding. In a broader context, the results support the hypothesis that allocating resources randomly leads police to catch more unlawful drivers. However, the strategy where police do not move from their assigned places is effective against drivers who try to speed aggressively (as it was the case with our dynamic-EWA agents).

Our model simplifies the domain to make the problem tractable. While this simpler model gives good insights into the dynamics of the interactions among agents, scalability to graphs of entire cities and much larger number of agents remains a challenge. A further challenge is to balance exploration and exploitation and guarantee a security value for the agents while remaining scalable. Our dynamic-EWA algorithm scales well to a reasonable amount of drivers and its path exploration can be improved by including paths that are some constant away from the optimal paths. However, it is not clear how driver agents will perform against opponents that can efficiently cover the graph.

Analysis of convergence and theoretical properties of the learning algorithms is a topic for future research. We will also consider models that capture other aspects of the domain, such as communication and potential collusion. Future research will investigate how agents can learn and adapt when police agents have different personalities, and will model risk-seeking and risk-averse drivers. Finally, we believe that our algorithm, analysis and models contribute positively towards inventing computational models that help police better allocate their limited resources, which is essential for better traffic control.

8. REFERENCES

[1] Baher Abdulhai, Rob Pringle, and Grigoris J. Karakoulas. Reinforcement learning for true adaptive traffic signal control. *J. Transp. Eng.*, 129:278–285, 2003.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.

[3] Ana L. C. Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multiagent Systems*, 18(3):342–375, June 2009.

[4] C. Camerer and T. Hua Ho. Experience-Weighted attraction learning in normal form games. *Econometrica*, 67(4):827–874, 1999.

[5] Colin Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, mar 2003.

[6] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, March 2008.

[7] Nicola Gatti. Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *Proc. 18th European Conference on Artificial Intelligence (ECAI)*, pages 403–407, 2008.

[8] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *J. Comput. Syst. Sci.*, 66:775–808, June 2003.

[9] Hiromitsu Hattori, Yuu Nakajima, and Toru Ishida. Agent modeling with individual human behaviors. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1369–1370, 2009.

[10] Teck H. Ho, Colin F. Camerer, and Juin-Kuan Chong. Self-tuning experience weighted attraction learning in games. *Journal of Economic Theory*, 133(1):177 – 198, 2007.

[11] Milind Tambe James Pita, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. GUARDS – game theoretic security allocation on a national scale. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, 2011.

[12] Dong-Hwan Kim and Doa Hoon Kim. A system dynamics model for a mixed-strategy game between police and driver. *System Dynamics Review*, 13(1):33–52, 1997.

[13] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. Nash in security games: an extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41(2):297–327, May 2011.

[14] Gergely Neu, András György, and Csaba Szepesvári. The online loop-free stochastic shortest-path problem. In *COLT*, pages 231–243, 2010.

[15] William T. B. Uther and Manuela M. Veloso. Generalizing adversarial reinforcement learning. Technical report, AAAI Fall Symposium on Model Directed Autonomous Systems, 1997.

[16] J. Wu, C. Ye, and S. Jin. Opponent learning for multi-agent system simulation. *Rough Sets and Knowledge Technology*, pages 643–650, 2006.