Electronic Preprint for Journal of Information Processing Vol.22 No.2

Regular Paper

Multiagent Decision Making on Transportation Networks

Ernesto Nunes[1,a)] Julio Godoy[1,b)] Maria Gini[1,c)]

Received: July 1, 2013, Accepted: January 8, 2014

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 destinations faster, while police units 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 using game theory. We describe the models and algorithms we developed and validate them on synthetic and real traffic data from different maps.

Keywords: game theory, adversarial reinforcement learning, Experience-Weighted Attraction, stochastic game

1. Introduction

Drivers who speed above the speed limit pose a threat to society, since they are more likely to cause traffic accidents with serious consequences [7], [19]. Therefore, catching drivers who speed is an important task for traffic control units, which deploy police cars and an increasing variety of sensors to slow down drivers and catch speeders. Despite the use of sensors, this remains a challenging problem for a variety of reasons: lack of knowledge of where drivers will speed, limited number of police units on patrolling tasks, and the ability of drivers to learn the location of police units and to adapt dynamically.

We model the problem of speed control as a general-sum repeated stochastic game [20] played by police units and drivers in a road network. Police units aim at preventing drivers from driving above the speed limit. Drivers speed up to reach their destination faster while trying to avoid getting caught.

Our choice of modeling the problem as a simultaneous game is motivated by the fact that drivers do not observe possible police locations prior to making their speeding decision (lack of observability), the fact that different drivers can simultaneously speed along different roads (multiple attackers) at any time (no fixed schedule), and that police units can move to different locations (mobile defenders). We model the game as a general-sum game since we do not assume that losses or gains of police agents are balanced by losses or gains of driver agents. We trade off the computational convenience of zero-sum games for a more general payoff relationship among agents. The probabilistic choice of actions of drivers and police units (probabilistic transitions) and the discounted rewards collected at each stage of the game makes stochastic games [20] a suitable model for our problem.

The game does not have pure strategy equilibrium points, and takes a long time to converge to a mixed-equilibrium [15]. Hence, we are interested in the dynamics during the game. We show empirically that police and driver agents can exploit each other when any of them makes a suboptimal choice.

This work makes the following contributions:

(1) We offer a general-sum stochastic game model for the speeding control problem.
(2) We propose an algorithm which combines learning principles from Experience-Weighted Attraction (EWA) [5] with dynamic programming. The algorithm is used by drivers to learn on what roads the chance of being caught while speeding is the smallest. The algorithm moves each driver from its starting to its destination locations and generates the drivers' speeding decisions. Police agents learn drivers' behaviors by observing their speeding decisions in the road network throughout the game.
(3) We present different algorithms for police units to decide where to position themselves to increase the probability of catching speeding drivers.
(4) We experimentally validate the model and algorithms using synthetic and real traffic data. The real data experiments were performed on data collected on road segments in London and nearby cities over a two months period.

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 complex with many heterogeneous agents, and can also be used to test other multiagent learning algorithms.

In Section 2 we review briefly the literature on patrolling, security, game theory, and machine learning. In Section 3 we define the game and review the learning principles of EWA and Opponent Q-learning. We present our agent design in Section 4. Experiments with synthetic traffic data and their results are presented in Section 5 and Section 6 respectively, while experiments and results with real traffic data are in Section 7 and Section 8. We discuss our findings in Section 9, and offer conclusive remarks

[1] University of Minnesota, Dept. of Computer Science and Engineering, Minesota, USA
[a)] enunes@cs.umn.edu
[b)] godoy@cs.umn.edu
[c)] gini@cs.umn.edu

and directions of future research in Section 10.

2. Related Work

Opportunities and challenges for the use of agents in traffic control are outlined in Ref. [4]. Most work in this area focuses on adaptive control of traffic lights (e.g., Ref. [1]) and intersections [6], or on modeling individual drivers' behaviors [11]. Kim et al. [15] 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 individual 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 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 Refs. [3], [9], [10] to patrol a fence or an area. There is a rich literature on modeling security problems as a Stackelberg game (e.g., Refs. [10], [21]). In this literature, the problem is modeled as a single leader (the police) and multiple followers (the attackers), where the leader deploys agents in different locations. This is the model we use for the real data experiments.

Jain et al. [13], [14] use a Stackelberg game formulation to schedule security patrols in road checkpoints in the Los Angeles International Airport to prevent terrorist attacks. A recent work by Fang et al. [8] extended the application of Stackelberg games to moving targets, allocating Coast Guard patrols to escort ferries. All these studies assume that the attacker can observe the strategy of the defender before acting. Instead, we model the problem as a simultaneous game because we do not assume drivers can observe police strategies prior to making decisions on speeding, and there are multiple attackers and multiple mobile defenders. These features make the use of Stackelberg leadership models not appropriate [16].

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, the arms correspond to the roads upon which agents make choices. The driver agents could plan their paths as in Ref. [18], in which loop-free stochastic shortest-paths are computed using multiarmed bandit solutions. Instead, in our case, the driver agents compute the shortest paths using Dijkstra's algorithm.

Our work benefits in particular from the literature on opponent learning [22] and EWA learning [5], [12].

3. Background

3.1 Game Definition and Assumptions

The game takes place on a network of roads represented by a weighted graph $G = \{V, E, W\}$. Each node $v_i \in V$ represents an intersection between two roads and each edge $e_i \in E$ represents a road. Each weight $w_i \in W$ represents the travel time through road segment e_i at the speed limit. We consider both directed and undirected graphs, to account for one- and two-way roads.

There are two types of players: driver and police agents. A driver agent has a starting location and a destination, both of them nodes in the graph. Driver agents keep on moving until they reach their destination. A police agent occupies one node in the graph at a time and is allowed to stay at the node or to move to an adjacent node. A game starts with all driver agents at their start locations, and ends when all drivers reach their destinations.

Nodes of the graph are abstracted into states. For the rest of the paper the terms nodes and states will be used interchangeably. In each state, agents have a discrete set of action choices. A driver agent has three choices: *not to speed*, to *speed up to 10 miles per hour* (*mph*) *over the speed limit*, or to *speed more than 10 mph over the speed limit*. A police agent can either *enforce a ticket* or *not enforce a ticket*.

We indicate the strategy space of player i as $S_i = \{s_i^1, s_i^2, \ldots, s_i^{m_i}\}$, where m_i is the number of strategies for agent i. A strategy profile is a combination on n strategies, one per player, $s = (s_1, s_2, \ldots s_n)$, where s_i is the strategy of player i. s_{-i} indicates a strategy profile for all player i's opponents.

A *pure strategy* in this game corresponds to the movement of the agent between two nodes and the deterministic choice of an action, which depends on the agent type. For a driver agent, the movement has to be to an adjacent node and the action is chosen from the set $\{\text{speed} \le L, L < \text{speed} \le L + 10, \text{speed} > L + 10\}$, where L is the speed limit. For instance, $s^1 = (v_3, \text{speed} \le L)$ is an example of a strategy for a driver agent who decides to go to node v_3 and not to speed, while $s^2 = (v_2, \text{speed} > L + 10)$ is a strategy for the same agent who decides to go to v_2 and to speed up.

A police agent either moves to an adjacent node or stays put and chooses to enforce or not a ticket. 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 enforces a ticket if a driver speeds. In s^2 the police moves to node v_2 and decides not to issue a ticket when it sees a speedy driver according to some probability. Placing probabilities over the actions allows us to model the choices real police have in giving tickets. In this work we set these probabilities, but they could be learned from domain experts or by mining data.

A *joint play* is the union of the simultaneous moves of drivers and police agents. After a joint play, all agents receive a payoff according to the payoff matrix (**Fig. 1**). We indicate the payoff function of player i as $U(s_i, s_{-i})$, since the payoff depends on the strategies chosen by the player and all of its opponents. We introduce the specific payoff functions we use in Section 4.

We assume the following *preferences*: (1) Driver agents prefer to speed more that 10 mph over the speed limit ($h_2 > g_2 > f_2$) if no police agent is predicted to be at the next node. (2) If

		Driver	
Police	speed $\le L$	$L <$ speed $\le L+10$	speed $> L+10$
E	a_1, a_2	b_1, b_2	b_1, c_2
N	a_1, f_2	g_1, g_2	g_1, h_2

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

police is predicted at the next node, drivers prefer not to speed ($a_2 > b_2$). (3) If drivers speed, police agents prefer to enforce a ticket ($b_1 > g_1$). Police agents enforce a ticket with higher probability if drivers speed more than 10 mph than if they speed less than 10 mph. Police agents get 0 payoff when drivers do not speed ($a_1 = 0$). The probabilities of enforcing tickets are defined in Section 5.2.

We assume that agents do not communicate among themselves. This means that police agents operate independently, without coordinating with each other. This assumption reduces the computational complexity of the learning process because it allows police agents to reduce their decision space to their local neighborhoods. However, this makes it impossible for police agents to coordinate their deployment and makes the problem harder to solve. To increase coverage in our experimental work we constrained police agents not to be at same node at the same time.

We assume that driver agents speed solely to shorten their travel time. Other factors that affect speeding decisions, for example emergency situations, can be modeled by changing the payoff values according to the importance agents place on each outcome. However, this is not a subject we explore in this work.

For learning purposes, we assume that both types of agents can access historical data of joint plays only for opponents in their neighborhood. This reduces the amount of information needed to make a decision. The choice does not affect the dynamics of the game, because we assume that police agents cannot give tickets to drivers they cannot see, and speeding driver agents do not stop speeding if there is no police on their current road.

3.2 Learning Algorithms

The learning principles of our algorithm for driver agents are inspired by the updating rules and soft-max action selection of EWA [5], combined with a Dijkstra-like algorithm to select the best road and speeding decision. An Opponent Q-learning [22] agent is used as a benchmark.

Experience-Weighted Attraction: Experience-Weighted Attraction [5] combines two learning models, belief and choice reinforcement. An agent maintains at each time period t its experience, $N(t)$, and its attraction to strategies, $A(t)$. Experience is measured as the discounted number of past experiences, where ϕ is the forgetting parameter, and κ the exploration parameter:

$$N(t) = \phi(1-\kappa)\cdot N(t-1) + 1, t \geq 1 \quad (1)$$

The attraction $A(t)$ to a strategy is computed using two parameters: ϕ, which is used to decay past attractions, and the attention parameter, δ, which represents the attention an agent pays to foregone payoffs. Both take values in [0,1]. If ϕ=1 the agent remembers all past plays, if ϕ=0 it forgets them.

The attraction to a strategy after time period t is the weighted sum of past attractions and payoffs (Eq. (2)). Agent i computes its attraction to strategy $s_i^j \in S_i$ as a combination of its prior experience, prior attraction, and the payoff collected from a joint play [12], where $s_{-i}(t)$ indicates the set of strategies used by all opponent players at time period t. I is an indicator function which returns 1 if s_i^j is the strategy chosen by agent i at time period t and 0 otherwise. The value of δ affects how much the not chosen strategies are reinforced. If $\delta = 1$ the attraction to all the strategies is reinforced by their full payoff, if $\delta < 1$ the attraction for the chosen strategy is reinforced by its full payoff but the attraction for the not chosen strategies is discounted by δ. The learning parameters ϕ and δ are replaced by functions that learn from historical data [12]. The definitions of these functions are presented in Section 5.2 and Section 7.2.

$$A_i^{s^j}(t) = \frac{\phi \cdot N(t-1)\cdot A_i^{s_j}(t-1)}{N(t)} + \frac{[\delta + (1-\delta)\cdot I(s_i^j, s_i(t))]\cdot U(s_i^j, s_{-i}(t))}{N(t)} \quad (2)$$

The probability that agent i chooses strategy s^j is computed using a logit function as

$$P_i^{s^j}(t+1) = \exp(\lambda \cdot A_i^{s^j}(t)) \Big/ \sum_{k=1}^{m_j} \exp(\lambda \cdot A_i^{s^k}(t)) \quad (3)$$

where the parameter λ measures the sensitivity of agent i to attractions. The agent plays a random response if $\lambda = 0$ or best responds if $\lambda = \infty$ [5].

Opponent Q-learning: Q-learning enables an agent to learn, as it acts and explores the environment, the optimal state-action value function $Q^*(v,a) = max_a Q(v,a)$, where $Q(v,a)$ is the expected discounted reward for executing action a in state v. The Q function is learned by repeatedly updating $Q(v,a)$ as follows:

$$Q(v,a) = Q(v,a) + \alpha[(R(v,a) + \gamma \max_a Q(v',a) - Q(v,a)] \quad (4)$$

where v' is the state reached by doing action a. α and γ are respectively the learning rate and the discount factor on future actions, and $R(v,a)$ is the immediate reward the agent receives for performing action a at state v.

Opponent Q-learning extends Q-learning by accounting for the presence of opponents [22] and improving over Minimax Q-learning [17]. As in Minimax Q-learning, Opponent Q-learning uses $Q(v,a,a_o)$ to indicate the Q-value of state v for action a from the agent and the joint action a_o of the opponents. It also assumes the opponent's behavior follows a stationary probability distribution (i.e., it is Markov) and hence it keeps track of how many times the opponent has chosen each action in each state. This is used to estimate the probability distribution of each opponent action in the state, $P(a_o|v)$. The probability is factored into the calculation of the expected value of $Q(v,a)$ by summing over the expected values of the joint plays:

$$E[Q(v,a)] = \sum_{a_o} P(a_o|v)Q(v,a,a_o) \quad (5)$$

In order to select the next state, the agent selects the action with the highest expected value, $\text{argmax}_a E[Q(v,a)]$. Accounting for the opponent's actions enables Opponent Q-learning to update the Q-values faster than Minimax Q-Learning does.

4. Design of Learning Agents

4.1 Environment Representation

On each edge of the road graph a driver has three speeding

choices. To represent the choices we augment the graph by replacing each edge in the original graph with three edges: a *non-speeding*, a *speeding up to 10 mph*, and a *speeding above 10 mph* edge. The weight of each edge equals the time to traverse it.

The payoff for traversing an edge depends on its weight and the outcome of the play. Let weight(s_i, v, v') be the weight of the edge connecting the agent's current node v to node v' using strategy s_i (i.e., the speeding decision). Let $H(v)$ be the weight of the heaviest non-speeding edge from v. The payoff driver agent i gets when going from v to v' using strategy s_i is defined as:

$$U(s_i, s_{-i}) = \begin{cases} 1 - \dfrac{\text{weight}(s_i, v, v')}{H(v)} & \text{if no ticket is received} \\ 0 & \text{if not speeding} \\ -1 & \text{if ticket received} \end{cases} \tag{6}$$

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

$$U(s_i, s_{-i}) = \begin{cases} 1 \text{ if driver speeds, is caught, and gets ticket} \\ 0 \text{ if driver speeds, is caught, but gets no ticket} \\ 0 \text{ if driver does not speed} \\ -1 \text{ if driver speeds, but there is no police} \end{cases} \tag{7}$$

4.2 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 of the algorithm (Algorithm 1) a driver agent follows two steps: (1) it computes a path using Dijkstra's shortest path algorithm assuming it will always speed and it chooses its strategy using EWA; (2) it updates its beliefs using EWA learning principles. The Dijkstra shortest path algorithm produces a lower bound on the travel time because it assumes drivers will always speed. However, the speeding decision is made using EWA (Eq. (3)), so the agent might not speed on all edges. Using Dijkstra's shortest path algorithm frees agents from having to learn which paths to follow, so agents can focus on speeding decisions. Another advantage is that drivers get loop-free paths at a relatively low computational cost.

Algorithm 1 Dynamic-EWA Driver Agent

```
1:  Initialize attraction, experience, and beliefs
2:  for each stage game do
3:      visited = ∅
4:      for v ∈ V do
5:          dist(v) = ∞
6:      current = start; dist(current) = 0; q = q ∪ (current, dist(current))
7:      while q not empty do
8:          current = argmin_v q    /* node with smallest dist */
9:          q = q − current
10:         visited = visited ∪ current
11:         for n ∈ neighbors(current) do
12:             if n ∉ visited then
13:                 dst = dist(current) + weight(current, n)
14:                 if dst < dist(n) then
15:                     prev(n) = current; dist(n) = dst
16:                     q = q ∪ (n, dst)
17:                     Choose strategy using Eq. (3)
18:     Compute the path using prev
19:     Play according to the path and chosen actions
20:     Update attraction, experience and probabilities using Eqs. (1)–(3)
```

The algorithm keeps a priority queue (q) to reduce computation time. We define dist(v) to be the distance of node v from the initial node (*start*), and weight(v, v') the weight of the edge from v to v', where v' is a neighbor of v. The algorithm computes the shortest path by updating the value of dist(v') whenever dist(v) + weight(v, v') is smaller than the current 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(v). Recovering the path entails starting from the goal node and following the parent node of each node found in prev(v) until the start node is reached. The EWA model is updated according to strategies on nodes in the path. The payoff for each edge is computed according to Eq. (6). Payoffs and historical data are used to compute the new values for Eqs. (1)–(3).

Analysis of the Algorithm:

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 the $N - 1$ neighbors of each node. Driver agents choose their strategies by evaluating the space of joint plays of the drivers and the entire police population. Since this space can be large, our algorithm makes two simplifying assumptions in order to reduce complexity: (1) a driver agent only needs to consider strategies of police agents in its neighborhood; (2) a driver agent is independent from other driver agents, hence it plans paths independently. The first assumption allows agents to prune the strategy space to consider. The size of the reduced strategy space is defined by the Cartesian product of the agent's strategies by the opponent's strategies, both constrained to the neighborhood of the current agent location.

The algorithm does not converge to pure strategy equilibrium points because such points do not exist in the game [15]. Driver agents drive above the speed limit if police agents do not enforce tickets. The increased number of speeding drivers will lead police agents to give tickets, which in turn will cause drivers not to speed. Hence, either agent type can increase its payoff by changing its strategy unilaterally. Intuitively, mixed equilibrium strategies are those for which the probabilities of police enforcing tickets make driver agents choose to speed or not with equal likelihood.

4.3 Opponent Q-learning Driver Agent Design

We designed Opponent Q-learning driver agents as a benchmark for Dynamic-EWA driver agents. The Q-values associated with each node in the road graph represent the discounted payoff of speeding or not 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 Eq. (4) 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 that 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 infer police presence when making a speeding decision.

5. Experiments with Synthetic Traffic Data

We designed a set of experiments with synthetic data to test the effectiveness of the agents in different environments and against different types of opponents.

5.1 Police Agents

We tested the following police agent types:

- *EWA-based*: to reduce the strategy space over which the agent has to sample, movements of these police agents are limited to a neighboring node. 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 5.2). These agents use Eqs. (1)–(3) to update their learning model.
- *Adaptive*: police agents can only move to a neighboring node. They analyze the payoffs in their current node and in neighboring nodes, and based on the number of observed speeding drivers, decide to move if a neighboring node seems more profitable in terms of the number of tickets they may issue.
- *Static*: police agents do not move to other nodes, they always stay in place.
- *Random*: police agents move randomly to any node in the graph. The only constraint that restricts their movement is the presence of another police agent at the node chosen, since no more than one agent can be at any node.

5.2 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 10 mph and below* to be 0.3. These values reflect the assumption that driving at higher speeds is more likely to cause accidents, hence the increased probability of receiving a ticket.

For the EWA-inspired algorithms, we found experimentally that $\lambda = 0.45$ for the driver and $\lambda = 0.65$ for 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 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 we use ($\kappa = 0.65$) 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 $\delta = \phi/2$. Hence, agents reinforce profitable strategies and depreciate strategies that lead to negative outcomes.

We found experimentally with the Opponent Q-learning drivers that $\alpha = 0.2$ gives more weight to newly computed reward values, allowing for faster convergence of the Q-values. Similarly, we found that $\gamma = 0.6$ weights the best neighboring Q-value so that better actions are preferred in earlier steps of the game than later. To prevent driver agents from going into a cycle, after a road is visited, we multiply its Q-values by a parameter (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.

5.3 Performance Metrics

In these experiments two performance metrics are used: the payoff driver agents accumulate during the iterations of the game and the regret driver agents experience. In our results we report smoothed payoffs and regrets. This helps to better see the trends in the average values of these quantities. Payoffs and regret do not have a specific unit. They solely represent agent preferences.

Let $U_{avg}(t)$ be the average payoff for driver agents at time period t (computed in Eq. (8)) and d be the number of drivers. To get smoother average payoff values and better see the overall trend of these values over time, we use a customized version of cumulative moving average (Eq. (9)). Let SA be the smoothed average payoff over β time periods for smoothing (in our experiments $\beta = 10$). We compute SA according to Eq. (9):

$$U_{avg}(t) = \frac{\sum_{i=1}^{d} U(s_i(t), s_{-i}(t))}{d} \tag{8}$$

$$SA(t) = \frac{SA(t-1) * \min(\beta, t) - 1 + U_{avg}(t)}{\min(\beta, t)} \tag{9}$$

The smoothed average payoff is computed recursively, starting with the average payoff for the first β time periods (where $SA(1) = U_{avg}(1)$), and computing the subsequent values using the value for the previous time period $SA(t-1)$ and the average payoff for the current period. When $t < \beta$, the smoothed average is computed over t periods ($\min(\beta, t)$).

Regret is measured as the difference between the payoff of the best possible strategy for a driver and the payoff the driver receives from the joint play. The highest regret on each edge has a value of 1 and occurs when a driver is caught speeding. Drivers who 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*, the best strategy in this case (s_i^{best}), and the payoff for speeding below 10 mph, the strategy actually chosen (s_i).

$$r_i(t) = U(s_i^{best}(t), s_{-i}(t)) - U(s_i(t), s_{-i}(t)) \tag{10}$$

$$r_{avg}(t) = \frac{\sum_{i=1}^{d} r_i(t)}{d} \tag{11}$$

$$SR(t) = \frac{SR(t-1) * \min(\beta, t) - 1 + r_{avg}(t)}{\min(\beta, t)} \quad (12)$$

Let $r(t)$ be the regret an agent experiences at time period t. In Eq. (10), s_i^{best} is the strategy that if chosen would yield the highest payoff, while s_i is the strategy the agent chose. Let SR be the smoothed average regret. The smoothed average regret (Eq. (12)) is computed in a similar manner as the smoothed average payoff.

We use the *maximum attainable average payoff* to compute the maximum amount of revenue driver agents can get in the absence of police. This value is used as an upper bound on the average payoffs drivers can earn. It is computed as the average travel time along paths where drivers speed more than *10 mph* above the speed limit.

6. Results with Synthetic Traffic Data

We show results on road 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 Minneapolis, which we call the Downtown graph.

6.1 Grid Graph

The Grid graph (**Fig. 2**) has 16 nodes and 72 edges, and contains loops. Twelve drivers start on the top two rows of the graph, and have different destinations in the bottom row. Six police agents are placed in the two bottom rows. We use synthetic traffic data and travel times. We ran 30,000 iterations per experiment.

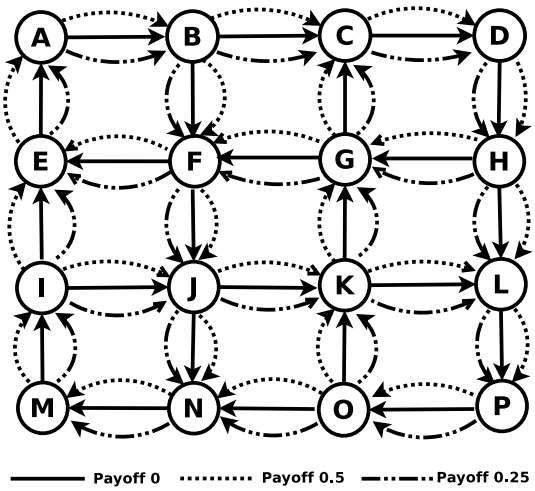


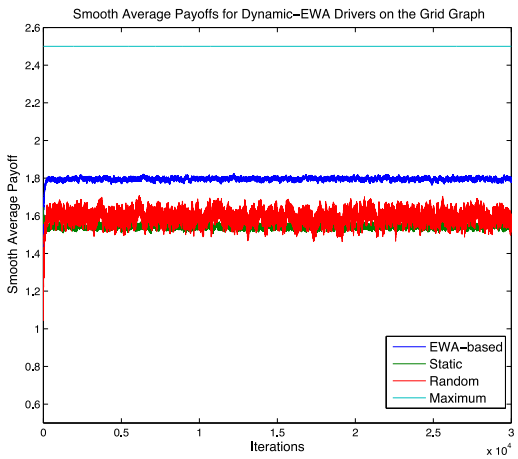
Fig. 2 Grid graph, with symmetrical weights on the edges. Edges between two nodes correspond to the three speeding choices for the drivers.

We use this graph to analyze driver and police agents behaviors when there are multiple paths to any destination.

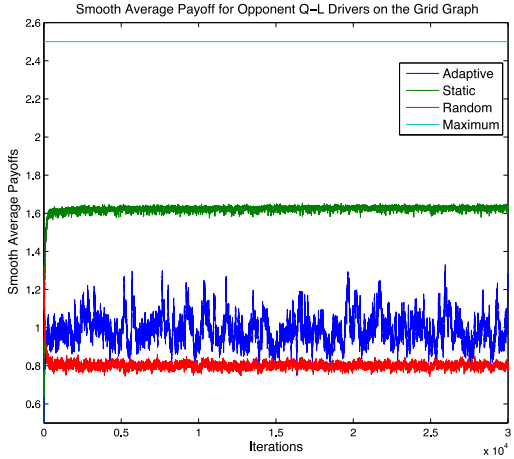
The rich set of alternative paths drivers can take in the Grid graph and their cost-symmetric nature (drivers attain the same cost regardless of which of these paths they choose) makes it a compelling case-study.

Results in **Fig. 3** (a) and Fig. 3 (b) illustrate an example in which adaptive driver agents perform poorly when playing against police agents that play randomly. The average payoff that both 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 the adaptive police agents, respectively. 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 a value of 1.6 (after smoothing the payoffs) when playing against random police. This suggest that randomness makes police locations unpredictable and hence it is the best strategy for police agents. This is consistent with results obtained when modeling security problems as Stackelberg games [21]. However, Stackelberg leaders play strategies with probabilities that maximize expected payoffs, while police agents playing randomly always play strategies with uniform probabilities, they do not attempt to maximize their payoffs.

The payoffs for dynamic-EWA drivers are roughly twice as large as the average payoffs of Opponent Q-learning driver agents (1.6 versus 0.8). The difference in average payoffs indicates that dynamic-EWA drivers adapt better against random police agents 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 actions, and hence agents adapt faster. Opponent Q-learning driver agents garner their highest payoffs when playing against

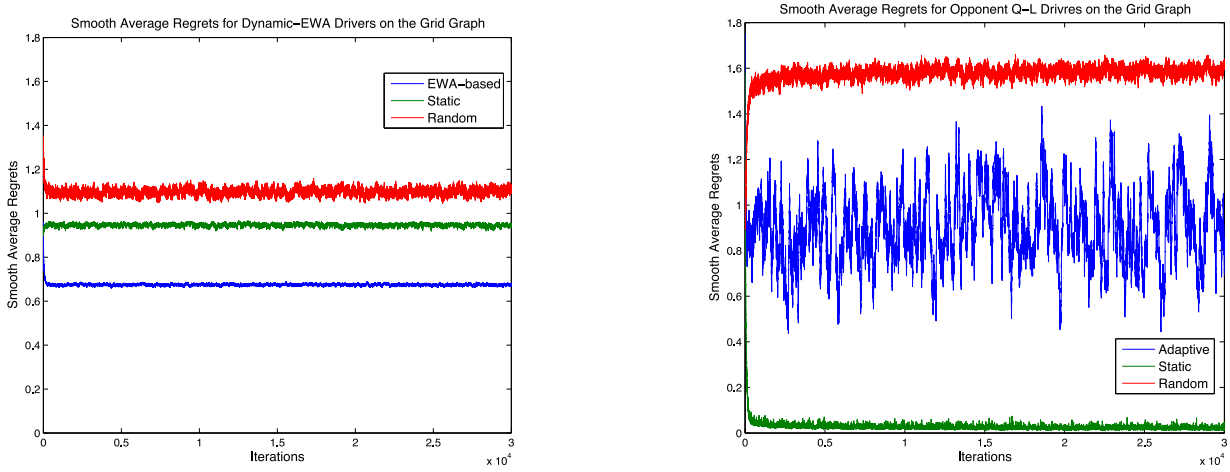


(a) Dynamic-EWA on the Grid graph



(b) Opponent Q-learning on the Grid graph

Fig. 3 Payoffs for dynamic-EWA and Opponent Q-learning drivers in the Grid graph.



(a) Dynamic-EWA on the Grid graph

(b) Opponent Q-learning on the Grid graph

Fig. 4 Regrets for dynamic-EWA and Opponent Q-learning drivers in the Grid graph.

static police agents (1.6). To the contrary, dynamic-EWA drivers perform poorly against static police. This is because EWA plays all strategies with non-zero probabilities, thus, dominated strategies may be probabilistically chosen. If instead the agent chose the best response every time, then, none of the dominated strategies would be chosen.

Dynamic-EWA drivers experience the highest regret when playing against police agents that play randomly. This is consistent with the payoffs agents receive. Opponent Q-learning drivers also experience highest regret when playing against police that play randomly (**Fig. 4** (b)), and smallest regret when playing against static police. Driver agents become conservative in the presence of random police and choose not to speed on roads in which they received tickets. Playing against static police illustrates how 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 against adaptive police is cyclic, evidencing adaptive behavior. Both dynamic-EWA and Opponent Q-learning drivers experience similar regrets (with the mean of the average regrets between 0.6 and 0.8) when playing against the EWA-based police and the adaptive police, respectively.

The large police to driver ratio (1:2, with 12 drivers and 6 police units) in this setup leads to low payoffs for driver agents. This example illustrates the fact that larger police presence leads drivers to lose more from suboptimal decisions.

6.2 Downtown Graph

The Downtown graph (**Fig. 5**) 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. Two of the three police agents start at locations that intersect drivers' shortest paths in at most one edge. This map was chosen to study how the size of the decision space affects the prediction ability of the agents.

In Fig. 5 we show the paths for two of the drivers, $D1$ and $D2$. The blue and green circles indicate their start and destination nodes respectively. 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 $P2$.

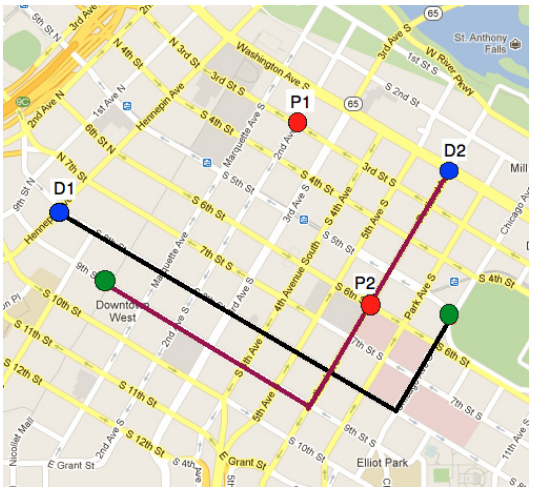
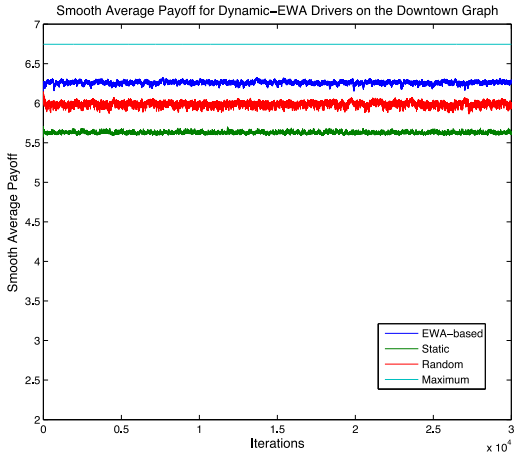


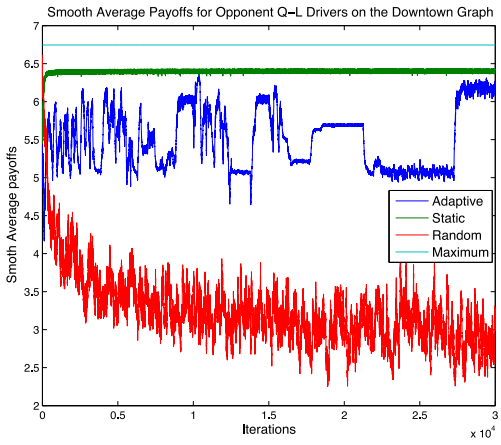
Fig. 5 Map of a section of the downtown of Minneapolis, with paths of two drivers and two police agents.

To generate traffic data we used real distances for the Downtown graph and computed the travel time assuming agents travel 30 mph. To compute the travel time when drivers speed, we randomly generated the travel times for speeding below and above 10 mph. We ran 30,000 iterations per experiment.

Driver agents that use Dynamic-EWA and Opponent Q-learning attain similar average payoffs when playing against EWA police and adaptive police (**Fig. 6** (a) and Fig. 6 (b)), respectively. The payoffs are close to the maximum attainable. The limited movement of both types of adaptive police agents combined with the large number of uncovered paths driver agents can take are the main reasons for the success of these driver agents. These police agents cover a smaller percentage of the area, which allows driver agents to speed without punishment on uncovered edges. The same is not true when playing against police agents that play randomly, since those agents can see the whole map. Dynamic-EWA drivers perform better than Opponent Q-learning drivers when playing against random police agents. This advantage is mostly

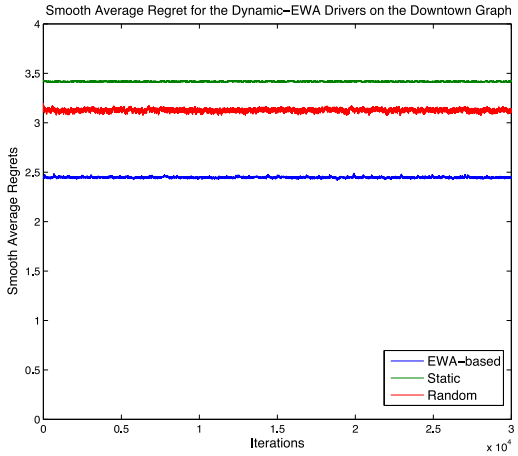


(a) Dynamic-EWA on the Downtown graph

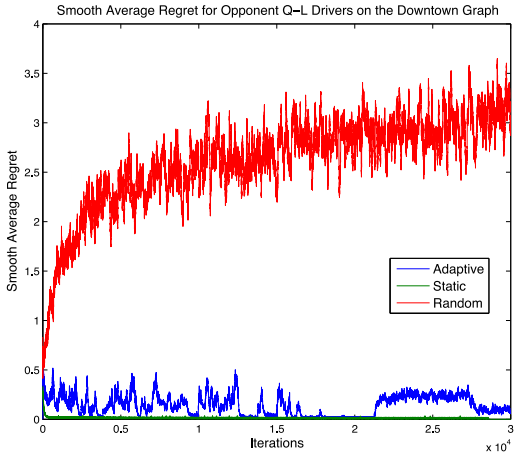


(b) Opponent Q-learning on the Downtown graph

Fig. 6 Payoffs for dynamic-EWA and Opponent Q-learning drivers in the Downtown graph.



(a) Dynamic-EWA on the Downtown graph



(b) Opponent Q-learning on the Downtown graph

Fig. 7 Regrets for dynamic-EWA and Opponent Q-learning drivers in the Downtown graph.

due to two factors: first, dynamic-EWA drivers only travel on the shortest paths returned by the Dijkstra's algorithm. Hence, agents explore fewer paths than Opponent Q-learning drivers. Exploring fewer paths makes it less likely for random police agents to occupy nodes on the driver's path more frequently than nodes outside the path. Thus, 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. 7** (a)) and for the Opponent Q-learning driver (Fig. 7 (b)) agents confirm that dynamic-EWA algorithms do worse against static police, while Opponent Q-learning drivers are able to learn to play against static police agents. It also confirms that both agents struggle against police playing random strategies.

7. Experiments with Real-Life Traffic Data

For our experiments with real data we use the traffic data generated in Motorways and A roads in the UK [*1] collected via speed sensors mounted on the monitored roads. A police agent uses these data and its learning algorithm to predict where drivers speed, and patrols these areas. The lack of ticket enforcement data, and the difficulties in inferring these events from the traffic data make it impossible to model driver speeding and routing choices in the road network. Therefore, we do not consider these experiments to be a game between police and driver agents, instead, it is a game between police and a nature player, where the strategies of the nature player are generated from the traffic data from driver agents.

Another factor that makes these experiments different from the ones with synthetic data is that we model police agents with a single centralized agent which deploys different police units. The centralized agent computes the location of the police units, so police units do not need to know the graph of the road network. Instead, only the central unit needs to know the current location of the agents, and allocate them according to the strategies that best-respond against the predicted driver speeding behavior in the network. This is convenient from a learning perspective, because only one agent learns from the data, instead of all the police units learning in a decentralized way.

[*1] We thank the UK Highways Agency and their partners for making this data available (http://data.gov.uk/dft-eng-srn-routes-journey-times).

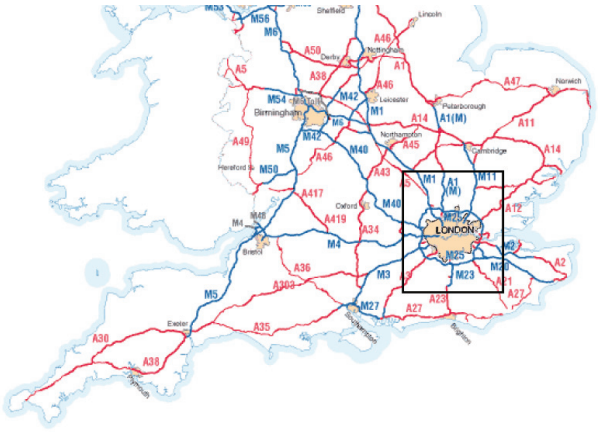


Fig. 8 Region from which links and traffic data were sampled.

7.1 Data Description and Preparation

The traffic data has many attributes of which we chose the following: *LinkId*, a unique identifier for each section of a road; *Date*; *Speed*, the average speed of cars entering a link over a 15 minute collection period; x and y location; and *Flow*, a measure of the number of cars that enter a link over the 15 minute period.

Due to the large number of links, we sampled the links by placing a bounding box over the map of road networks of London and nearby cities and collecting all the links that have one end within the box. The corners of the box have the following latitude and longitude coordinates: (51.75, −0.586), (51.75, 0.086), (51.46, −0.586), (51.46, 0.086), respectively (roughly shown in the black box on the map in **Fig. 8**). Our choice of the area is motivated by the concentration of links (4,960 links) and the high volume of traffic data in the region, which generates enough data for the learning algorithm. To further reduce the amount of data, we sampled data for 10 and 30 links, respectively.

There are 96 daily traffic data points for each link. The data points are generated from aggregating the data from all the 15 minute intervals in the day. To generate *speeding events*, we average the speed on each link over the 96 data points and compare the average to the speed limit for the link (70 mph or 112 kilometers/hour). Speeding occurs if the average is above the speed limit. To simplify the generation of speeding events, we assume that all vehicles and all roads have the same speed limit. The assumption on the speed requirements for the roads is reasonable since most roads of interest are motorways and A roads, which have a speed limit of 70 mph.

We use two months of data (December 2012 and January 2013). The data in the first month are used to generate prior probabilities and historic data of drivers' speeding choices. The data are also used to tune the learning parameters for the police agents. The data in the second month are used to measure agents performance (model evaluation period).

7.2 Police Agents

We tested the following centralized police agent types:

- *EWA-Trained*. This agent uses, as the EWA police agents in the synthetic case, Eqs. (1)–(3) to update its learning model. The difference is that this agent starts with prior probabilities on the links to patrol, which are learned from the drivers average speeding in the training data. Moreover, the agent starts with a nonempty history, because the data for the first month are used to compute where drivers sped.
- *EWA-Untrained*. This agent differs from the trained one because it starts with uniform probabilities on the links to patrol (no prior knowledge). However, like the trained agent, it uses the learning parameters tuned using the training data.
- *Random*. This agent has no specific knowledge on where drivers are likely to speed, so it allocates the units with uniform probabilities over the links.

The probabilities used as prior for the learning model of the EWA-Trained police agent are computed as the average across the probabilities of speeding on each link over the entire training month. The same data was used with a randomized grid search to tune the EWA parameters.

While most of the parameters remain the same as in the experiments with synthetic data, we used a different function to compute the parameter ϕ.

Let $h(t)$ be the entire history the EWA police agent observes for all links, and $h^l(t)$ the history for link l. In the history the agent keeps track of the frequency with which drivers sped in each link from the beginning of the training data up to time t.

Let $rh(t)$ be the recent history at time t. The recent history is a window over the last N days, where $N \leq |h(t)|$. N is a parameter to our learning algorithm, and its optimal value is computed during parameter tuning. The value for the history of a link is 1 if drivers always sped on the link throughout the length of history (both for the recent history and the entire history), and 0 if drivers never sped on the link. The recent and entire history values for a link are numbers in the range $[0, 1]$.

Using the history variables we compute the value of ϕ at time t in the same way as Ho et al. [12], where L is the set of links:

$$\phi(t) = 1 - \frac{1}{2}\sum_{l=1}^{L}(h^l(t) - rh^l(t))^2 \qquad (13)$$

In addition to the EWA parameters κ, λ and W, we also find the optimal value for the minimum speeding probability in a link (ϵ). This is a threshold value that allows police to consider links with low probabilities of speeding as links where speeding does not occur. The grid search keeps the set of values for the parameters for which the average police regret is minimized.

8. Results with Real-Life Traffic Data

We performed two sets of experiments, one with 10 road links and five police units, and another with 30 links and 15 police units. We report:

- The *average payoff* per police unit that the central police agent collects in a day. This is reported over the evaluation period. Unlike the payoff definition used in Section 4, here the payoff is proportional to the number of roads where speeding occurred and where police agents were allocated. A payoff of 1 indicates that a police unit was correctly allocated to a road link with speeding; a payoff of −1 indicates that a police unit was allocated to a link where speeding did not occur while links where speeding occurred remained un-

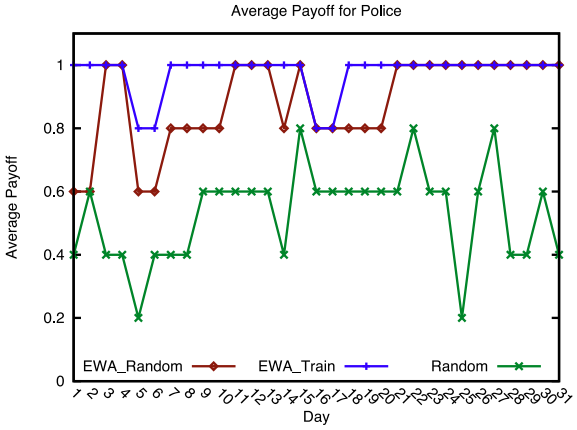


Fig. 9 Average payoffs per police unit per day for the two EWA police types and the Random police agent with 10 links, 5 police units. Parameter values used are $\kappa = 0.01$, $\lambda = 0.5$, $\epsilon = 0.0061$ and $W = 7$.

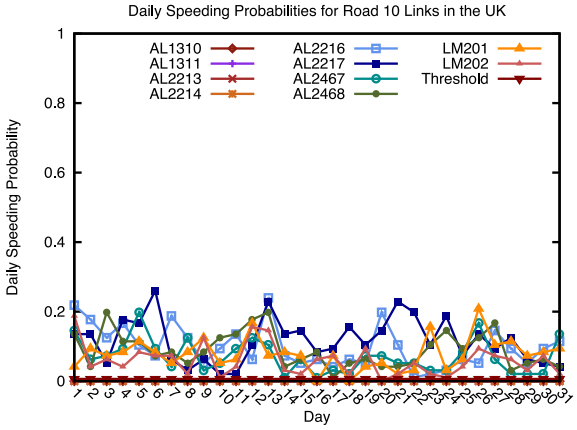


Fig. 10 Daily probabilities of speeding on 10 links during the month of data used for evaluation. The threshold speed is set to 0.0065 ruling out 40% of the links.

covered; a payoff of 0 indicates that a police unit was allocated to a link where speeding did not occur and speeding did not occur in any other uncovered link. The average payoff per police unit is computed as in Eq. (8). The maximum average payoff per police unit in a day is 1 because we assume that each unit monitors one road per day.

- The *daily probabilities of drivers speeding* in the sampled links, The daily probabilities of speeding for a link are computed by dividing the number of periods in which the average speed in the link was above the speed limit by the total number of 15 minute intervals (96 intervals in a day).
- The *percentage of links* in which speeding occurs.
- The *police allocation efficiency*. The police allocation efficiency is computed as the proportion of police units correctly allocated to links in which speeding occurs. This was computed only for the EWA-Trained police agent, but the EWA-Untrained police agent has similar values.

Figure 9 shows the average payoff for police for the experiment with 10 links and five units. The results show that the EWA-Trained police agent (line with crossed patterns) predicts better or with the same accuracy as the other types of police agents. EWA-Trained superiority is due to seeding. This agent's beliefs on where drivers are likely to speed are seeded with probabilities that summarize the historic speeding behavior of drivers in the past month. Meanwhile, the EWA-Untrained agent does not have the initial knowledge, hence it starts with random probabilities of patrolling links and it learns until it attains the maximum payoff, at which point it places all the agents in roads in which the probability of speeding is high.

Note however that even when police plays randomly there are many days (e.g., 9–13 and 16–21 in Fig. 9) in which their average payoff is 0.6, indicating that the five police units were able to cover three out of the five links they could cover. This is due to the high daily percentage of links where speeding occurs (60%), which is shown in **Fig. 10**. The daily percentage is the percentage of roads in which speeding occurred at any time in a given day.

The higher is the number of links where speeding does not occur, the poorer the prediction ability of police units becomes. This is shown in **Fig. 11** where the 15 police units that have to cover 30 links predict poorly when there are many roads in which speeding

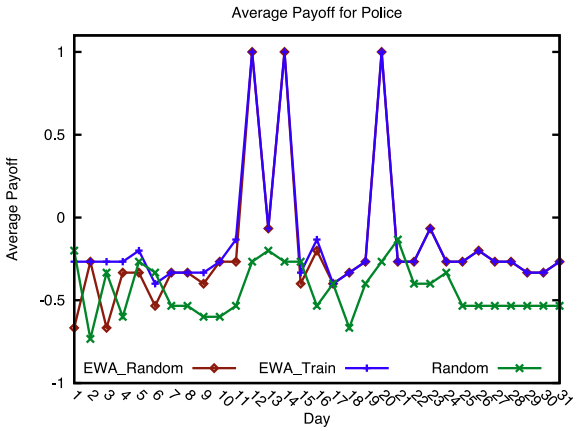


Fig. 11 Average payoff per police unit per day for the two EWA police types and the Random police agent with 30 links and 15 police units. Parameter values used are $\kappa = 0.1$, $\lambda = 0.5$, $\epsilon = 0.01$ and $W = 7$.

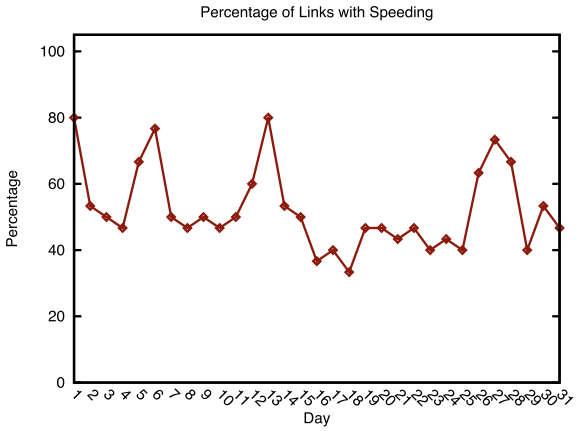


Fig. 12 Percentage of the 30 links in which speeding occurred during the month of data used for evaluation.

does not occur and therefore receive negative payoffs.

Figure 12 shows the daily percentage of speeding for the experiments with 30 links and 15 police units. There are four peaks: the first one is on the 1st day (January 1st 2013), the second on the 6th, the third on the 12th, and the fourth on the 27th day. In some cases the EWA police agent is able to identify these peaks. For example, it takes advantage of the higher percentage of speeding in the links on the 12th and 14th days to fully use its units (100% as shown in **Fig. 13**), thus collecting the maximum possi-

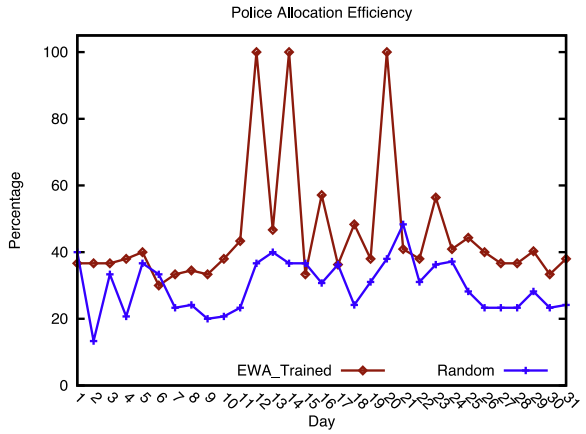


Fig. 13 EWA-Trained versus Random Police allocation efficiency. Experiment with 30 links and 15 units per agent.

ble average payoff. However, it mis-predicts the high speeding percentage for both the 13th and 27th days. On the other hand, the police agent playing randomly mis-predicts continuously and ends up with negative payoffs throughout the entire learning period.

Figure 13 provides evidence that the EWA-Trained police agent has higher prediction power than the random police agent. This result can also be extended to the EWA-Untrained agent. As shown in the figure, the EWA police agent exploits the high percentage of speeding on the links of the road network, while the random agent does not. The advantage EWA police enjoy is attributed to learning, which helps the agent mitigate its losses.

9. Discussion

The interactions between police and driver agents in our experiments with synthetic traffic data show that both types of driver agents experience higher regret when playing against police agents that play randomly than they do when playing against police agents that learn. Random police agents are not rational players, hence playing against them leads to situations from which drivers cannot learn. The frequency with which random police agents change strategies, and their disregard for payoff maximization is at the heart of the difficulty drivers experience when predicting police behavior. Therefore, while this type of police can take advantage of the rational decisions drivers make, hence the high driver regret, adopting their strategies does not lead to equilibrium.

Compared to Opponent Q-Learning drivers, our results show that EWA drivers attain higher payoffs when playing against random police agents than Opponent Q-Learning drivers. We argue that the fast detection of changes in opponent strategies is the main feature of EWA that leads to this result. In each learning step, drivers revisit history to see which strategies opponents used, and if all strategies were uniformly chosen, driver reinforce them with similar experience values. Thus, the only factor that differentiates strategies is the payoff for each strategy. In the worst-case, the probabilities EWA drivers compute will be nearly uniform across strategies, depending on the negative outcomes when playing different strategies.

A more surprising result is that drivers that use the dynamic-EWA algorithm experience more regret when playing against police agents that do not move from their assigned positions. This is attributed to the fact that on average, EWA agents play all strategies with non-zero probabilities (unless equilibrium is achieved). This seems consistent with our understanding of speeding behaviors of human drivers. Whenever human drivers get a ticket on a road they might stay some time without speeding, but eventually they try again, in which case a fixed police agent will ticket them again. Contrarily, Opponent Q-Learning drivers exploit the higher police predictability and experience lower regrets against static police than EWA drivers do.

Our experiments with real traffic data show that learning from driver-generated traffic data helps police better predict drivers' speeding behavior and leads to higher average payoffs than when playing with random strategies. These experiments differ from the ones with synthetic data because drivers do not respond to the strategy choices of the police agents. Instead, police agents mine the traffic and speeding data to build models of driver speeding behavior, and play against these models. The main disadvantage of such an approach is that the drivers' reaction to police strategies cannot be accurately measured, hence, the driver speeding model we build might not conform to the actual strategic choices drivers would make if they interacted directly with our police agents. Therefore, we cannot directly compare our results here with our results for the synthetic experiments. However, we can still conclude that learning when and where driver agents speed does improve police allocation efficiency.

10. Conclusions and Future Work

We modeled and built a full simulation of a stochastic game involving police and driver agents. We proposed the dynamic-EWA algorithm that enables drivers to adapt to police agents. Our algorithm shows that the agents learn to adjust their strategies according to the opponent strategy. We performed further experimental studies with real traffic data from the road network of London and nearby cities in which police used sampled data to learn when and where speeding occurred. The main finding of the latter experiments is that the EWA police agents learn, although not perfectly, and outperform a police agent that allocates its units randomly.

Future work involves theoretical analysis of the convergence properties of the dynamic-EWA algorithm and possibly more complex models to capture other aspects of the domain, such as communication and potential collusion, and the effect of signs like "Speedometer" or "Radar Enforced" on the roads in the model.

Another interesting avenue to further our studies and make the system more applicable is to train the system offline by learning from both ticket enforcement and traffic data. EWA police agents would learn how drivers respond to ticket enforcement events, and build more accurate models of driver behavior. Such a system could then be deployed online, and continue to learn as it interacts with speed sensors and enforcement data.

References

[1] Abdulhai, B., Pringle, R. and Karakoulas, G.J.: Reinforcement Learning for True Adaptive Traffic Signal Control, *J. Transp. Eng.*, Vol.129,

pp.278–285 (2003).
[2] Auer, P., Cesa-Bianchi, N. and Fischer, P.: Finite-time Analysis of the Multiarmed Bandit Problem, *Mach. Learn.*, Vol.47, No.2-3, pp.235–256 (2002).
[3] Basilico, N., Gatti, N. and Amigoni, F.: Leader-follower strategies for robotic patrolling in environments with arbitrary topologies, *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pp.57–64 (2009).
[4] Bazzan, A.L.C. and Klügl, F.: *Introduction to Intelligent Systems in Traffic and Transportation*, Morgan & Claypool (2013).
[5] Camerer, C. and Ho, T.H.: Experience-Weighted Attraction Learning in Normal Form Games, *Econometrica*, Vol.67, No.4, pp.827–874 (1999).
[6] Dresner, K. and Stone, P.: A Multiagent Approach to Autonomous Intersection Management, *Journal of Artificial Intelligence Research*, Vol.31, pp.591–656 (2008).
[7] Elvik, R.: Speed and Road Safety: Synthesis of Evidence from Evaluation Studies, *Journal Transportation Research Record: Journal of the Transportation Research Board*, Vol.2005, pp.59–69 (2005).
[8] Fang, F., Jiang, A.X. and Tambe, M.: Optimal patrol strategy for protecting moving targets with multiple mobile resources, *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pp.957–964 (2013).
[9] Gatti, N.: Game Theoretical Insights in Strategic Patrolling: Model and Algorithm in Normal-Form, *Proc. 18th European Conference on Artificial Intelligence* (*ECAI*), pp.403–407 (2008).
[10] Gottlob, G., Leone, N. and Scarcello, F.: Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width, *J. Comput. Syst. Sci.*, Vol.66, pp.775–808 (2003).
[11] Hattori, H., Nakajima, Y. and Ishida, T.: Agent modeling with individual human behaviors, *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pp.1369–1370 (2009).
[12] Ho, T.H., Camerer, C.F. and Chong, J.-K.: Self-tuning experience weighted attraction learning in games, *Journal of Economic Theory*, Vol.133, No.1, pp.177–198 (2007).
[13] Jain, M., Pita, J., Tambe, M., Ordóñez, F., Paruchuri, P. and Kraus, S.: Bayesian Stackelberg games and their application for security at Los Angeles international airport, *SIGecom Exchanges*, Vol.7, No.2, Article 10 (2008).
[14] Jain, M., Pita, J., Tsai, J., Kiekintveld, C., Rathi, S., Ordóñe, F. and Tambe, M.: Software Assistants for patrol planning at LAX and Federal Air Marshals Service, *Interfaces*, Vol.40, No.4, pp.267–290 (2010).
[15] Kim, D.-H. and Kim, D.H.: A system dynamics model for a mixed-strategy game between police and driver, *System Dynamics Review*, Vol.13, No.1, pp.33–52 (1997).
[16] Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V. and Tambe, M.: Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness, *Journal of Artificial Intelligence Research*, Vol.41, No.2, pp.297–327 (2011).
[17] Littman, M.: Markov games as a framework for multi-agent reinforcement learning, *Proc. Int'l Conf. Machine Learning* (1994).
[18] Neu, G., György, A. and Szepesvári, C.: The online loop-free stochastic shortest-path problem, *COLT*, pp.231–243 (2010).
[19] Nilsson, G.: Traffic Safety Dimensions and the Power Model to Describe the Effect of Speed on Safety, PhD Thesis, Lund University (2004).
[20] Shapley, L.S.: Stochastic Games, *Proc. National Academy of Sciences*, Vol.39, No.10, pp.1095–1100 (1953).
[21] Tambe, M.: *Security and Game Theory: Algorithms Deployed Systems, Lessons Learned*, Cambridge University Press, New York, NY (2011).
[22] Uther, W.T.B. and Veloso, M.M.: Generalizing Adversarial Reinforcement Learning, Technical Report CMU-CS-03-107, Carnegie Mellon University (2003).

Ernesto Nunes is a Ph.D. candidate in Computer Science at the University of Minnesota. He received a B.A. in Computer Science from Macalester College in 2010 and a M.S. in Computer Science from the University of Minnesota in 2013. His interests are in multiagent cooperation and learning, and applied game theory.

Julio Godoy is a Ph.D. candidate in Computer Science at the University of Minnesota. He received a B.S. in Computer Engineering in 2004 and a M.S. in Computer Science in 2008 from the University of Concepción, Chile. His interests are in multiagent learning and game theory.

Maria Gini is a Professor in Computer Science and Engineering at the University of Minnesota. She is on the editorial board of numerous journals, including Artificial Intelligence, Autonomous Agents & Multi-Agent Systems, Robotics and Autonomous Systems, Web Intelligence and Agent Systems. She is a Fellow of AAAI, and a Distinguished Scientist of ACM. Her interests are in autonomous robots, economic agents, distributed allocation of tasks, team work, and learning opponent behaviors.