Multi-Agent Evolutionary Game Dynamics and Reinforcement Learning Applied to Online Optimization of Traffic Policy

Yuya Sasaki

Department of Economics, Utah State University, 3530 Old Main Hill, Logan, UT 84322-3530

Phone: (435) 797-8217 *Email*: yuyas@econ.usu.edu

Fax: (435) 797-3944

Nicholas S. Flann

Department of Computer Science, Utah State University, 4205 Old Main Hill, Logan, UT 84322-4205

Phone: (435) 797-2432 *Email*: nick.flann@usu.edu

Fax: (435) 797-3265

Paul W. Box

Commonwealth Scientific and Industrial Research Organization, Heath Road, Alice Springs, NT 0870, Australia.

Phone: (08) 8950-7100 *Email*: paul.box@csiro.au

Fax: (08) 8950-7187

Multi-Agent Evolutionary Game Dynamics and Reinforcement Learning Applied to Online Optimization of Traffic Policy

Abstract

This chapter demonstrates an application of agent-based selection dynamics to the traffic assignment problem. We introduce an evolutionary dynamic approach that acquires payoff data from multi-agent reinforcement learning to enable a adaptive optimization of traffic assignment, provided that classical theories of traffic user equilibrium pose the problem as one of global optimization. We then show how this data can be employed to define the conditions for evolutionary stability and Nash equilibria. The validity of this method is demonstrated by studies in traffic network modeling, including an integrated application using geographic information systems applied to a complex road network in the San Francisco Bay area.

Keywords**:** agent-based, evolutionary games, geographic information systems, reinforcement learning, transportation.

INTRODUCTION

When we think about alternative routes to drive from an origin to a destination, there are several factors of travel time to consider, such as the distances, speed limits, and possible congestion. Although people appear to have an incentive to use the shortest-distance routes, this does not happen in reality because the supply function of traffic roads exhibits an increasing cost nature. In other words, we have monotonically increasing travel time with respect to increasing traffic flow volume.

Assume a simple example where two agents travel from an origin to a destination, with only two paths available, a short path and a long path. Also assume that the long path has greater capacity with more lanes, than the shorter path. The supply function of the longer path has a flatter slope and a higher intercept (free-flow travel time) than the short path. Suppose that the two agents (denoted by A and B) make decisions simultaneously. Then the outcome cost matrix for the four possible pure-strategy combinations will look like Table 1. If both choose the greedy strategy (the short path), the consequence is a tragedy where both get trapped in severe congestion. The non-cooperative equilibria in this pure strategy setting are thus the symmetric pair of (A:1 B: 2) and (A: 2 B: 1) in Table 1. Whenever some agents choose the shortest paths, others have to compromise to inferior choices. The characteristics of traffic behaviors illustrated

Time Matrix (larger values are worse)		B's Action	
		short path	*long path*
A's Action	*short path*	A: 5 B: 5	A: 1 B: 2
	long path	A: 2 B: 1	A: 3 B: 3

Table 1. Example payoff matrix of four combinations of choices. The values are travel time in any time unit.

by this example will be analyzed in more general form in the succeeding sections.

Notations and Definitions of Traffic Networks

A traffic network consists of nodes and arcs, where an arc is always closed by a pair of nodes. We use the term “O-D pair” to refer to a pair of origin and destination nodes in an agent’s trip. A path is an ordered sequence of arcs that connects two nodes that are not necessarily adjacent to each other. Thus, an O-D path refers to an ordered sequence of arcs that connects the origin and destination nodes. Let *I*, *J*, and *R* denote sets of arcs, nodes, and paths in a network, respectively. *R* must be defined for each O-D pair, while *I* and *J* can be global for all the O-D pairs. Since we will focus first on the case of one O-D pair (later in Sec 4.2 we will consider multiple origins and one destination), we will not use any subscript or superscript to *R*. Let $I_j \subseteq I$ denote the set of arcs radiating from node $j \in J$, where I_j also represents the set of available strategies for agents being at node *j*. An arc is a component of the network in the entire system, while it can also be perceived as a “strategy,” “next move,” or “action” from decision-makers’ viewpoint. We will exclusively use the term “arc” when the network structure is discussed, while the term “strategy” interchangeably with “arc” will be used when we discuss agents’ decision-making.

Classical Theory of Traffic User Equilibrium

Before discussing the main idea, let us review classic theory of equilibrium states in traffic networks. Let **v** denote the vector of arc flow volumes for all elements in *I*, and let **w** denote the vector of path flow volumes for all elements in *R*. Correspondingly, let **t**(**v**) denote the vector of arc travel times as a function of arc flow volumes, and let **u**(**w**) denote the path travel times.

Additionally, we use flag variable $\delta_{i,r}$, which is 1 if path *r* includes arc *i*, and 0 otherwise. Given these, Beckmann et al. (1956) introduced the following optimization problem, the solution to which yields deterministic user equilibrium for a fixed travel demand *d*.

$$\min\ f(\mathbf{w}), \qquad \text{where } f(\mathbf{w}) = \sum_{i\in I}\int_0^{v_i} t_i(x)dx.$$

While *f* may seem unrelated to **w**, $\sum_{r\in R} w_r\delta_{i,r}$ can be substituted for v_i for all *i*. This problem is subject to the constraint of flow volume conservation

$$g(\mathbf{v}) = 0, \qquad \text{where } g = d - \sum_{r\in R} w_r .$$

Thus, setting up Lagrangean $L(\mathbf{w},\lambda) = f(\mathbf{w}) - \lambda g(\mathbf{w})$, we obtain the following Kuhn-Tucker first-order conditions (the second-order conditions follows the monotonicity assumption on $t_i(v)$).

$$w_r \frac{\partial L(\mathbf{w},\lambda)}{\partial w_r} = w_r(u_r(\mathbf{w}) - \lambda) = 0 \quad \forall r \in R,$$

$$\frac{\partial L(\mathbf{w},\lambda)}{\partial w_r} = (u_r(\mathbf{w}) - \lambda) \geq 0 \quad \forall r \in R, \text{ and}$$

$$\frac{\partial L(\mathbf{w},\lambda)}{\partial \lambda} = d - \sum_{r\in R} w_r = 0 \quad \forall i \in I .$$

The second condition implies that Lagrangean multiplier λ equals the minimum path travel time among all the paths in *R*. With this and the first condition, the path flow amount of *r* can be positive only if its path travel time equals the minimum path travel time. Intuitively, only those paths that bring about the minimum travel time in the network can attract vehicles. This basically extends the engineering principle proposed by Wardrop (1952), which will be further extended by multi-agent evolutionary dynamics, as discussed in the next section.

MULTI-AGENT GAME WITH REINFORCEMENT LEARNING

Model Description

Define a payoff matrix $\mathbf{A}_j$ for each node $j \in J$, where $\mathbf{A}_j$ is square and the number of columns and rows corresponds to the number of elements in I_j. Each element $a_{ii'}$ of $\mathbf{A}_j$ is a marginal payoff of flow on arc *i'* for those on arc *i*. Assume that agents being at node *j* have perfect information of the traffic volumes of all the arcs in I_j. Let $\mathbf{x}_j$ denote the vector representation of those volumes for node *j*. Provided that we only deal with one destination case, the value of $\mathbf{e}_i\mathbf{A}_j\mathbf{x}_j$ represents an estimated travel time to be spent from node *j* to the destination node provided that the agent chooses arc $i \in I_j$ as an available option at node *j*. Hence, agents should take strategy *i* such that $i = \arg\min_{\bar{i} \in I_j} \mathbf{e}_{\bar{i}}\mathbf{A}_j\mathbf{x}_j$ to minimize the total travel time from node *j* to the destination. Recalling the Bellman equation (Bellman, 1957), we can find the relationship between this game approach and the conclusion of the classical theory discussed in the previous section. If agents are rational,

$$\mathbf{e}_{\bar{i}}\mathbf{A}_j\mathbf{x}_j = t_{\bar{i}} + \min_{\tilde{i} \in I_{\bar{j}(\bar{i})}} \mathbf{e}_{\tilde{i}}\mathbf{A}_{\bar{j}(\bar{i})}\mathbf{x}_{\bar{j}(\bar{i})} \tag{1}$$

holds from Bellman equation where $\bar{j}(\bar{i})$ is the succeeding node's index as a result of taking strategy $\bar{i}$ at node *j*. Thus

$$\begin{aligned}
\lambda &= \min_{i \in I_{Origin}} \mathbf{e}_i\mathbf{A}_{Origin}\mathbf{x}_{Origin} \\
&= \min_{i \in I_{Origin}} (t_i + \min_{\bar{i} \in I_{j(i)}} \mathbf{e}_{\bar{i}}\mathbf{A}_{j(i)}\mathbf{x}_{j(i)}) \\
&= \min_{i \in I_{Origin}} (t_i + \min_{\bar{i} \in I_{j(i)}} (t_{\bar{i}} + \min_{\tilde{i} \in I_{\bar{j}(\bar{i})}} \mathbf{e}_{\tilde{i}}\mathbf{A}_{\bar{j}(\bar{i})}\mathbf{x}_{\bar{j}(\bar{i})})) \\
&\quad \vdots
\end{aligned}$$

where node $\bar{j}(\bar{i})$ succeeds node $j(i)$ through arc $\bar{i}$, which succeeds the origin node through Arc *i*. An important implication is that the problem of $\min_{\bar{i}\in I_j} \mathbf{e}_{\bar{i}}\mathbf{A}_j\mathbf{x}_j$ at each node is equivalent to agents choosing only those paths with the least travel time, an obvious conclusion from backward recursion. Of course, a restriction arises that all of these formulations hold only if $\mathbf{A}_j$ is accurately defined for all *j*. When this is not true, agents will still have the incentive and ability to modify $\mathbf{A}_j$ to make it more accurate based on their experiences.

We formulate this process of modification of $\mathbf{A}_j$ through experience using reinforcement learning (Sutton, 1988; Sutton and Barto, 1998), an artificial intelligence approach. While we have treated traffic volumes as elements of state space, let us now assume that the total volume departing from node *j* is constant in a unit time interval and that state $\mathbf{x}_j$ is contained by a simplex so that the model will be compatible with multi-agent evolutionary game theory. Given these assumptions, let $\Theta_j = \{\mathbf{x}_j \in \mathbb{R}_+^n \mid \sum_{\bar{i}\in I_j} x_{j,\bar{i}} = 1, \mathbf{x}_j \geq 0\}$ be the simplex for node *j*, where *n* denotes the number of elements in I_j. Then, set I_j defined in Sec 1.1 can be considered as a pure strategy set since $I_j = \{\bar{i} \in \mathbb{N}^1 \mid e_{\bar{i}} \in \Theta_j\}$. Let $\Omega = \{(\bar{i}, \mathbf{x}_j) \mid \bar{i} \in I_j, \mathbf{x}_j \in \Theta_j\}$ be the set of (*n*+1)-tupple parameters of strategy-state pairs, and $\Lambda_j = \{(Q_j(\bar{i}))_{n\times 1} \mid \bar{i} \in I_j\}$ be the set such that $Q_j(\bar{i})$ is the agent's estimate of travel time, $\mathbf{e}_{\bar{i}}\mathbf{A}_j\mathbf{x}_j$. Note that set Λ_j specializes in Node *j* for one destination. An agent performs decision making by using the modifiable function $F_j : \Omega_j \to \Lambda_j$. Though there exist many ways to realize this function, let us adopt a discretized state model or tabular state space. Suppose that Θ_j is partitioned into mutually disjoint subsets such that $\bigcup_l \Theta_j^l = \Theta_j$ and $\Theta_j^l \cap \Theta_j^m = \varnothing$ for all $m \neq l$. If the model employs **v** rather than **x**, a similar discretization on the set of **v** must be carried out instead (see Sec 4.2).

Define the new set $\overline{\Omega} = \{(\bar{i}, l) \mid \bar{i} \in I_j, \Theta_j^l \subset \Theta_j\}$, and with a discrete indexing by *l*, the former function can be rewritten as

$$\overline{F}_j : \overline{\Omega}_j \rightarrow \Lambda_j . \qquad (2)$$

This simplification by discretization is employed for the sake of computational simplicity. From computational viewpoint, this is a simple tile coding technique with no overlapping cells. We employed the resolution of ten tiles for each variable. For example, this allows the function ($\overline{F}_j$) to be represented by such simple forms as arrays or linked lists whose elements are easily modified by reinforcement learning. The function (*F*F_j) could be defined continuously, but this would require more complex learning methods which employ neural networks as function approximators described in Tesauro (1995). As previously described, the decision process employing (2) is defined as $i^* = \arg\max_{\bar{i}} \overline{F}_j(\bar{i}, l)$.

Since the states are discretely defined, let $Q_j(i,l)$ denote the estimated value returned by the function $\overline{F}_j(i,l)$, thus we have the identity $\mathbf{e}_i \mathbf{A}_j \mathbf{x}_j \equiv Q_j(i,l)$ as a result of the completion of learning. Especially for the equilibrium state $\mathbf{x}_j^*$, this identity

$$\mathbf{e}_i \mathbf{A}_j \mathbf{x}_j^* \equiv Q_j(i, l^*) \qquad \text{where } l^* = l' \, s.t. \, \mathbf{x}_j^* \in \Theta_j^{l'} \qquad (3)$$

becomes an important resource in our analysis, as shown later. Note again that this variable specializes in node *j* for one destination. Suppose an agent having departed node *j* reaches the destination node, with his actually experienced travel time between these two nodes given by R_j. Then, the error of $Q_j(i,l)$ given by $R_j - Q_j(i,l)$ can be used by Monte Carlo learning to update $Q_j(i,l)$ is defined as

$$Q_j(i,l) := \frac{1}{t}\left[R_j - Q_j(i,l)\right] + Q_j(i,l), \qquad (4)$$

where *t* is the number of times this update has occurred including the current time. This equation defines $Q_j(i,l)$ as the equally-weighted average from all the past experiences. This function can be approximated by substituting a constant $0 \le \alpha \le 1$ for 1/*t* in (4) thereby obtaining a learning method where the value of $Q_j(i,l)$ is a weighted average, where more recent errors are weighted higher:

$$Q_j(i,l) := \alpha\left\lfloor R_j - Q_j(i,l)\right\rfloor + Q_j(i,l), \tag{5}$$

that is more effective for stochastic environment than (4). This parameter, α, may be associated with what Roth and Erev (1995) referred to as the degree of forgetting. Recall that our original purpose was to find the accurate representation of $\mathbf{A}_j$ for better estimates of $\mathbf{e}_i\mathbf{A}_j\mathbf{x}_j$. Finding the closed form unique solution to this is feasible only if exactly *n* unique values of R_j are obtained for each row of $\mathbf{A}_j$, thus making a system of *n* linear equations for *n* unknowns for each row. But it is very unlikely that this condition holds for real or simulated environments. Hence we replace this closed form approach by (5) (and later (6) and (7)).

As an extension to (5), Sarsa and Q-learning are the instances of temporal difference learning, which uses an incremental backup method in updating *Q* values. In temporal difference learning, the estimated travel time, R_j, at node *j* succeeded by node $\bar{j}$ is computed as $R_j := t_{\bar{i}} + Q_{\bar{j}}(\cdot)$, where $t_{\bar{i}}$ denotes the travel time that the agent actually spent on arc $\bar{i}$ between node *j* and $\bar{j}$, a temporal difference. The Sarsa algorithm, an online version of this, is formulated as follows.

$$Q_j(i,l) \Leftarrow Q_j(i,l) + \alpha[t_i + Q_{\bar{j}}(\bar{i},\bar{l}) - Q_j(i,l)], \tag{6}$$

where $\bar{l}$ is the index used to identify the state $\mathbf{x}_{\bar{j}}$. This updates $\overline{F}_j$ for the element of (*i*,*l*) in $\overline{\Omega}_j$.

If node $\bar{j}$ succeeding node *j* is the destination node, then Monte Carlo update rule (5) is substituted for (6). Likewise, the offline version called Q-learning (Watkins, 1989) is formulated as follows.

$$Q_j(i,l) \Leftarrow Q_j(i,l) + \alpha[t_i + \max_{\bar{i}} Q_{\bar{j}}(\bar{i},\bar{l}) - Q_j(i,l)], \qquad (7)$$

which resembles equation (1), implying that (7) improves learning by using Bellman optimality equation. Again, if node $\bar{j}$ succeeding node j is the destination node, then Monte Carlo update rule (5) is substituted for (7).

From replicator equation (Taylor and Johnker, 1978), $(\log x_{j,i}\dot{)} = \mathbf{e}_i \mathbf{A}_j \mathbf{x}_j - \mathbf{x}_j \mathbf{A}_j \mathbf{x}_j$, we derive $(\log x_{j,i}\dot{)} - (\log x_{j,\bar{i}}\dot{)} = \mathbf{e}_i \mathbf{A}_j \mathbf{x}_j - \mathbf{e}_{\bar{i}} \mathbf{A}_j \mathbf{x}_j$ which implies the equilibrium condition

$$\mathbf{e}_i \mathbf{A}_j \mathbf{x}_j^* - \mathbf{e}_{\bar{i}} \mathbf{A}_j \mathbf{x}_j^* = 0 \quad \forall i, \bar{i} \in I_j .$$

Provided that estimates $\mathbf{e}_i \mathbf{A}_j \mathbf{x}_j \equiv Q_j(i,l)$ are sufficiently accurate, then we may translate the above equilibrium condition to the following definition.

Definition 1. *The interior equilibrium condition for multi-agent reinforcement learning is defined as*

$$Q_j(i,l^*) = Q_j(\bar{i},l^*) \qquad (8)$$

$\forall i, \bar{i} \in I_j$ *where* i *and* $\bar{i}$ *have strictly positive flow volumes.*

Additionally, we apply the Nash equilibrium condition, $\mathbf{x}_j^* \mathbf{A}_j \mathbf{x}_j^* \le \mathbf{x}_j \mathbf{A}_j \mathbf{x}_j^* \ \forall \mathbf{x}_j \in \Theta_j$, to argue the following claim. (Notice that the Nash equilibrium condition usually has the reverse inequality sign as $\mathbf{x}_j^* \mathbf{A}_j \mathbf{x}_j^* \ge \mathbf{x}_j \mathbf{A}_j \mathbf{x}_j^* \ \forall \mathbf{x}_j \in \Theta_j$. But we have $\le$ instead because we are *minimizing* travel time for optimality in our model, rather than maximizing.)

Claim 1. *Nash equilibrium condition for multi-agent reinforcement learning is equivalent to*

$$\sum_{i\in I_j}(x^*_{j,i}-x_{j,i})Q_j(i,l^*)\le 0 \quad \forall \mathbf{x}_j \in \Theta_j\,. \tag{9}$$

(And for usual setting, inequality $\ge$ *replaces* $\le$ *in (9).)*

Proof. Nash equilibrium condition, $\mathbf{x}^*_j\mathbf{A}_j\mathbf{x}^*_j \le \mathbf{x}_j\mathbf{A}_j\mathbf{x}^*_j \ \forall \mathbf{x}_j \in \Theta_j$, can be written as

$\sum_{i\in I_j} x^*_{j,i}\cdot \mathbf{e}_i\mathbf{A}_j\mathbf{x}^*_j \le \sum_{i\in I_j} x_{j,i}\cdot \mathbf{e}_i\mathbf{A}_j\mathbf{x}^*_j \ \forall \mathbf{x}_j \in \Theta$, which by identity $\mathbf{e}_i\mathbf{A}_j\mathbf{x}^*_j \equiv Q_j(i,l^*)$ of (3) turns out to be equivalent with what we desire to find. *Q.E.D*

We may define the condition for evolutionarily stable state (ESS) for multi-agent reinforcement learning in a similar way. ESS condition in our setting, where smaller cost is better, is defined such that $\mathbf{x}^*_j\mathbf{A}_j\mathbf{x}_j < \mathbf{x}_j\mathbf{A}_j\mathbf{x}_j$ holds for all $\mathbf{x}_j \ne \mathbf{x}^*_j$ in the neighborhood of $\mathbf{x}^*_j$ in Θ_j.

Claim 2. *ESS condition for multi-agent reinforcement learning is for*

$$\sum_{i\in I_j}(x^*_{j,i}-x_{j,i})Q_j(i,l)<0\,. \tag{10}$$

to hold for all $\mathbf{x}_j \ne \mathbf{x}^*_j$ *in the neighborhood of* $\mathbf{x}^*_j$. *(And for usual setting, inequality* $>$ *replaces* $<$ *in (9).)*

Proof. The proof is the same as that of Claim 1, except that $Q_j(i,l)$ replaces $Q_j(i,l^*)$ and that the strong inequality sign replaces weak inequality. *Q.E.D*

These definitions and claims are helpful in analyzing the dynamic behavioral properties of games based on simulation results of agent-based computational models, as shown in the next section.

A Note about Social Learning

The model described, in fact, includes social learning (Bandura, 1977) mechanism as the shared policy $\{Q_j\}$ is used and updated by arbitrary agents at each time step. This is the form of shared knowledge with heterogeneous learning, as opposed to heterogeneous knowledge, which must be distinct in properties. This social learning approach is beneficial especially if the knowledge space is too broad for an individual to cover alone, and sharing of knowledge by a group will mitigate the burdens of learning about the world, which would be laborious otherwise. As the network is complicated, this feature will become not only useful, but almost necessary. However, we must note its drawbacks. As we will see later in Sec. 4.1, shared knowledge can mislead the others' behaviors once atypical reinforcements wrongly influence the public knowledge.

EXAMPLE OF BEHAVIORAL ANALYSIS BY SIMULATION

Environment Description

We have developed a simple agent-based computational model to simulate and analyze the game behaviors of traffic policy. In this simplified environment, there is only one O-D pair, consisting of two paths connecting the O-D, and each path includes only one arc. Hence, there are no more than 2 options available in the network. This kind of state abstraction is common, for example in

Arc ID	*lsmt*	t^0	*leng*
1	200	90	202
2	400	45	190

Table 2. Parameters used for behavioral analysis simulation.

analyses of commutation between urban center and suburban zonal center. The equation for arc travel time is given by a linear monotonic function of arc flow volume as $t_i = t_i^0 + (lsmt_i / leng_i) \cdot v_i$, where $leng_i$ is the arc length, $lsmt_i$ is the length-scaled marginal travel time, and t^0 is zero-flow travel time. The concrete values of parameters are given in Table 2. Arc 1 is longer but has a larger capacity. In other words, Arc 2 is faster if the traffic volume is small while arc 1 will become faster as traffic volume increases. One or more artificial agents, in which Monte Carlo learning (5) is embedded, are instantiated at each discrete time step at the origin node. Agents travel on Arc *i* at the speed of $leng_i/t_i$. One instance of $\overline{F}$ (and thus one set of $Q(i,l)$) is shared by all the agents in order to realize faster social learning.

In fact, this simple model is capable of being examined by analytics, by identifying mixed-strategy equilibria in a population context for varying levels of travel demand. One can indeed see that a unique interior equilibrium exists for a fixed demand. A purpose of our employment of computational model here is to confirm that the policy function built by the artificial agents with learning ability does not contradict with this analytic inference. Besides, the extensions to be discussed in the next section suggest that analytical approaches which would involve game behaviors with bootstrapped value functions would be difficult.

Results

After a run of 20,000 time steps, we obtained a set of $Q(i,l)$ and thus $\overline{F}(i,l)$ for each *i* and *l*. A policy graph of $Q(2,l) - Q(1,l)$ plotted over (v_1, v_2) state space is given by Fig 1. Since we varied the number of agents departing the origin node, we need to deal with **v** rather than **x**. (Note that we do not have subscript *j* to **x**, since we have only one decision node in this model. So, just as in usual vector notation, let subscript *k* of x_k represent the index of the component of vector **x**, rather than node index.) However, a general normalization of **v** in a subspace $\{\mathbf{v} \mid |\mathbf{v}| = c\}$ for a constant *c* translates to **x**. Hence, we will not lose generality. In Fig 1, there exists a subset of state space expressed as $\{\mathbf{v} \mid Q(2,l) - Q(1,l) = 0\}$, which by definition (8) represents the set of equilibria. Path 1 has greater travel time to the left of this equilibrium set, and Path 2 has greater travel time to the right of it. It is only along the equilibrium set where agents are indifferent between choosing Path 1 and Path 2.

In order to make a game approach, let us take an example of *c* = 150. The same policy graph plotted over this subset of the state space is given by Fig 2. Geometrically, this graph is a cross-sectional image of Fig 1 in the diagonal direction. Since we know $|\mathbf{v}| = 150$, we can normalize **v** to **x**. It has a point at $\mathbf{x}^* \approx (0.65, 0.35)$ where policy geometry is crossing the horizontal line of $Q(2,l) - Q(1,l) = 0$. This is the equilibrium in this subspace, as defined in (8). For this equilibrium to be Nash equilibrium, $(0.65 - x_1)Q(1,l^*) + (0.35 - x_2)Q(2,l^*) \leq 0$ must hold for all $\mathbf{x} \in \Theta$ from (9). Since *n* = 2 or $x_2 = 1.0 - x_1$, this condition can be simplified to $(0.35 - x_2)(Q(2,l^*) - Q(1,l^*)) \leq 0$, which is obviously satisfied because of $Q(2,l^*) - Q(1,l^*) = 0$. Hence we conclude that $\mathbf{x}^* \approx (0.65, 0.35)$ is also Nash equilibrium. Additionally, (10) is requiring that $(0.65 - x_1)Q(1,l) + (0.35 - x_2)Q(2,l) < 0$ must hold for all $\mathbf{x} \neq \mathbf{x}^*$ in a ball around $\mathbf{x}^*$ for this equilibrium to be an ESS. Again, by the same logic as what we used for Nash

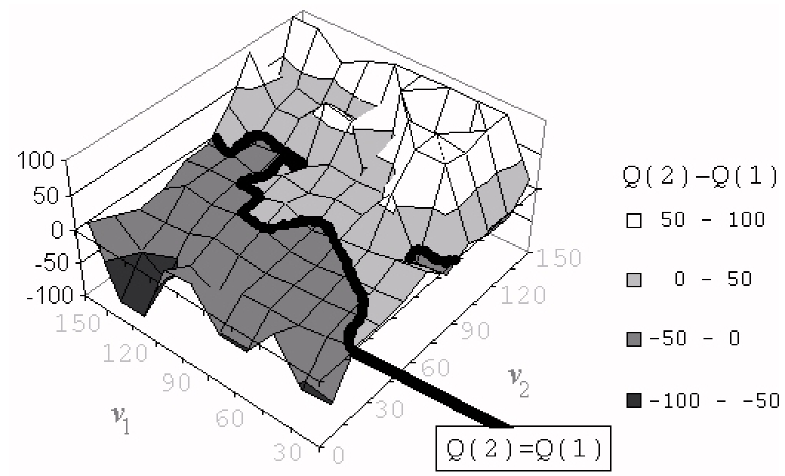


Figure 1. Policy graph of $Q(2,l)-Q(1,l)$ for two-path network.

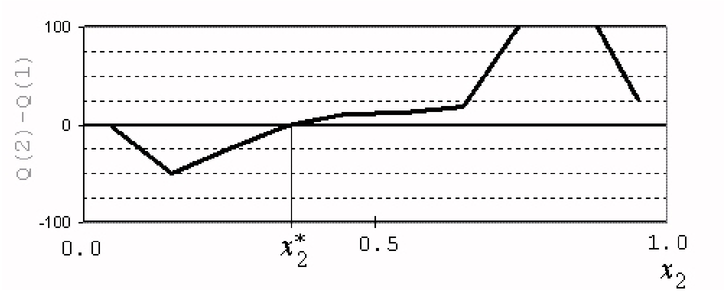


Figure 2. Policy graph of $Q(2,l)-Q(1,l)$ plotted over 1-dimensional subspace $\{\mathbf{v} \,|\, |\mathbf{v}|=150\}$.

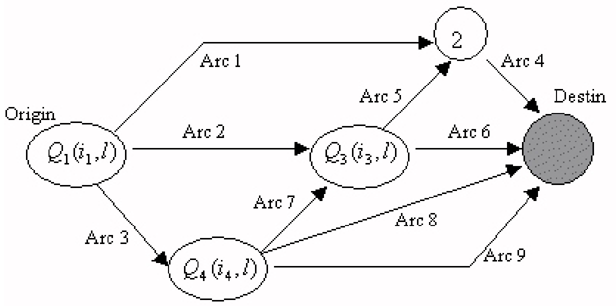
equilibrium identification, this condition can be simplified to $(0.35-x_2)(Q(2,l)-Q(1,l))<0$, with which we can easily conclude from Fig 2 that $\mathbf{x}^* \approx (0.65, 0.35)$ is also an ESS. Actually, the fact that the geometry of Fig 1 is formed such that $Q(2,l^*)-Q(1,l^*)<0$ to the left of equilibrium line and $Q(2,l^*)-Q(1,l^*)>0$ almost everywhere (and completely everywhere in the ball around $\mathbf{x}^*$) to the right of the equilibrium implies that all the elements in the set of equilibria are also in a set of ESS. This result shows that the dynamics has a continuous asymptotically stable set bounded by the border of Θ for this particular example, provided that the Q geometry (Fig 1) generated by agents through learning is sufficiently accurate.

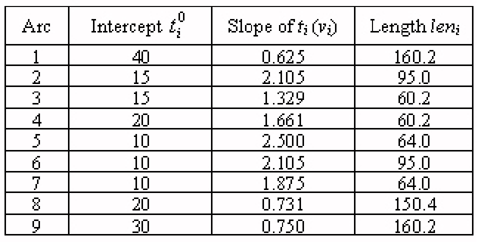
In this section, we showed a role that agent-based computational models play in helping our theoretical analyses for a relatively complicated dynamic system such as traffic network, based on the definition and claims discussed in the previous section. Simulation results of multi-agent reinforcement learning showed that the model is capable of generating some resources from which we may derive equilibria, NE, and ESS.

DEMONSTRATIONS

A Simple Traffic Network

In this section, we use a simple network (Fig 3) to examine the effectiveness of the three reinforcement learning algorithms (5)-(7) in the traffic policy optimization problem. The network is a directed graph connecting one pair of origin and destination by 9 arcs and 3 intermediate nodes. There are 7 paths in total in this network. Arc 4 is designed as a congested road, for 3 paths out of all the 7 paths use this arc. In contrast, Arc 8 and Arc 9 are designed with flatter-sloped supply functions common in high capacity roads. Agents leave the origin for the destination. At every time step, one agent is instantiated and put on origin node. There are 3 nodes at which agents make decisions. These nodes are marked with ellipsoidal symbols including the notation $Q_j(i, l)$, representing the value of the function $\overline{F}_j$ (Note that node 2 is not a





Arc	Intercept t_i^0	Slope of $t_i(v_i)$	Length len_i
1	40	0.625	160.2
2	15	2.105	95.0
3	15	1.329	60.2
4	20	1.661	60.2
5	10	2.500	64.0
6	10	2.105	95.0
7	10	1.875	64.0
8	20	0.731	150.4
9	30	0.750	160.2

Figure 3. Graphical representation of simple traffic network and the attribute data of arcs.

decision node, since the solely available strategy there is $i = 4$). This value stores the estimated travel time to be spent from node j to destination given state l and strategy i. We expect that intelligent agents learn to avoid Arc 1 and Arc 5 as well as Arc 4, since these deterministically lead to the congested arc in the directed graph. On the other hand, they may learn to choose Arc 3 since it leads to Arc 8 and Arc 9.

The deterministic traffic user equilibrium presented in Sec. 1.2 implies zero variance of travel times among all the used paths at the optimum. The top row of Fig 4 shows the variances acquired from the simulation of 20,000 iterations with Monte Carlo learning (5) for $\alpha = 0.1$. Additionally, we define the exploration probability $\varepsilon = 0.3$ at which agents take non-optimal action $i' \neq i^*$. (The need for exploration comes from the need to define A_j accurately *for each row* in our original problem.) In the initial phase, the variances are high, reflecting the inefficient traffic flows. This occurs because agents do not have sufficient knowledge at first, and have to go through explorations of the undiscovered world. Once the exploration period is over, the state continues to approach the equilibrium until the end of simulation. The learning rate of $\alpha = 0.1$ was chosen because of its relative superiority. When the learning rate is set too high Q_j values change too much based on each learning iteration and the resulting Q_j values mislead the agents. On the other hand, when the learning rate is too low the learning cannot “catch up” with the dynamics of the environment, and thus the modified Q_j values are likely to be obsolete. Hence, moderate values of α such as 0.1 work best. This is true not only for Monte Carlo control, but also for Sarsa and Q-learning.

The second and the third rows of Fig 4 show the variances generated by Sarsa and Q-learning, respectively. The results that use temporal difference seem inferior to Monte Carlo control for this special case of simple network, both in the closeness to equilibrium and stability.

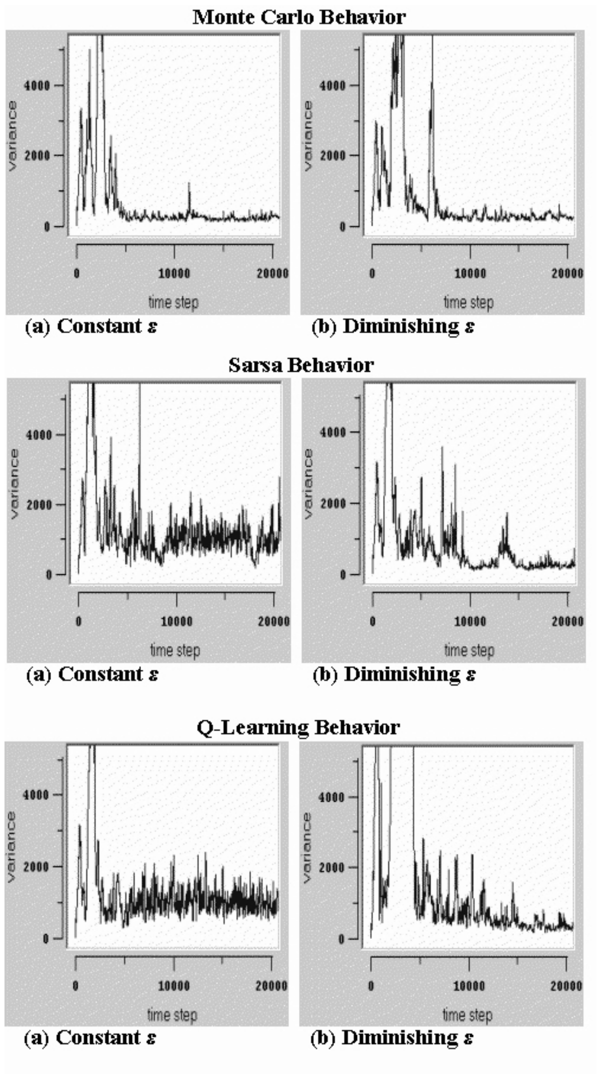


Figure 4. Variances of travel times among seven paths for each of three reinforcement learning algorithms.

This sort of experimental result is rare since temporal difference learning usually improve the updates of *Q* values in many tasks (Sutton and Barto, 1998). This unusual result may be attributed to the unique property of multi-agent model sharing one set of *Q*. For example, agents having traveled on Arc 2 updates $Q_1(2, l)$ value based on $Q_3(i', l')$ using (6) or (7), but this update depends on what strategy *i'* these agents have chosen and the state *l'* they have perceived.

If some agents chose to explore a non-greedy action such as *i'* = 5, where Arc 4 proceeded by Arc 5 is very congested, then the update to $Q_1(2, l)$ based on $Q_3(5, l')$ will be biased toward an overestimated value. While some agents update such a biased values of $Q_{j'}$, other agents simultaneously use this biased $Q_{j'}$ to backup other Q_j values, and this spiral of misdirected backups severely harms social knowledge. One way to lessen this disadvantage is to set the exploration probability ε as a diminishing variable instead of a constant. We tested this with a simple rule defined by the difference equation $\Delta\varepsilon / \Delta time = -0.001\varepsilon$. The variance generated with decreasing ε is shown in right side of Fig 4. This method gradually decreases the magnitude of oscillations for Sarsa and Q-learning.

We also tested the same agent-based simulation where the agents make random choices instead of using reinforcement learning, which only resulted in variances remaining hundred thousands throughout simulation. In contrast, reinforcement learning brought about emergent traffic equilibrium, with variances around or even below 1,000. The variance of 300 by Monte Carlo control, for example, means that the standard deviation of path travel times are only 17 time steps, where 120 is the average path travel time. We see that the paths have close travel times with reinforcement learning. Specifically, simulation results with variances close to zero characterize rough convergences to equilibria; the equilibrium condition as given by Definition 1 requires the equality of the travel times for all the used path entailing a variance of zero.

Demonstration with a GIS Network Data

In this section, a demonstration of the algorithm in a little more sophisticated network than the previous section is examined. Some special-purpose tools such as geographic information systems (GIS) become necessary when a data of actual traffic network is concerned. One idea is to load GIS data, instantiate each geographic object, and then put artificial agents in the modeled geographic space. Though most GIS data are given in table (database) format, employment of

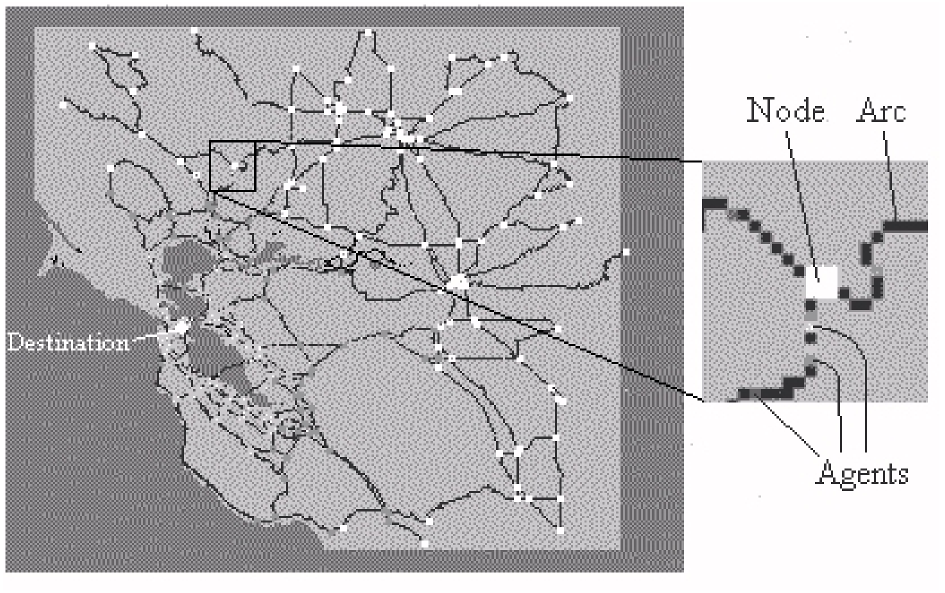


Figure 5. Traffic network of San Francisco Bay area.

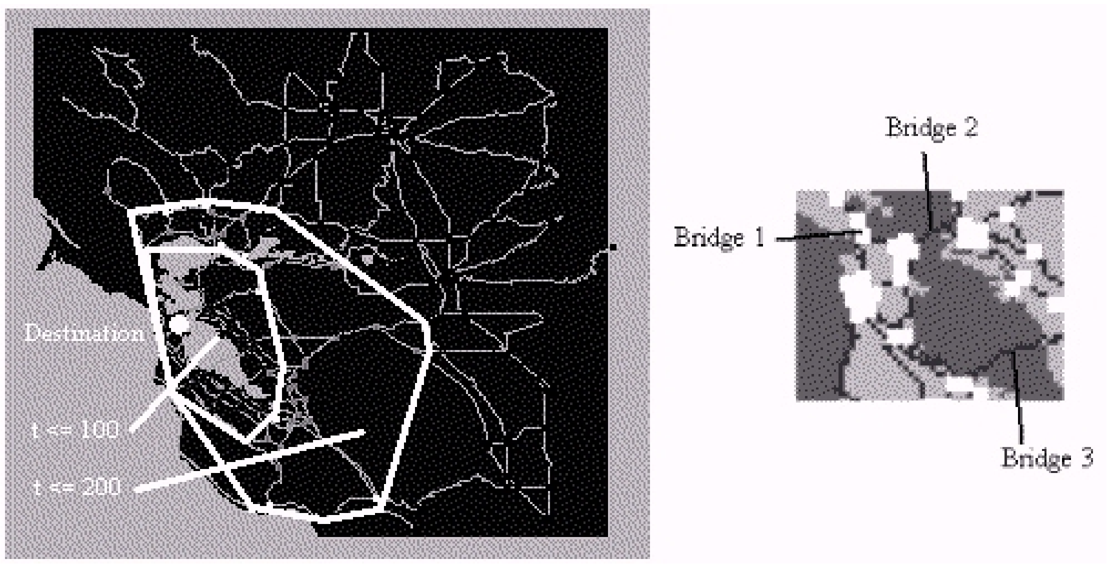


Figure 6. Convex hulls of nodes within 100 and 200 time steps of travel time to destination (left); and three main bridges leading to the destination (right).

object-oriented graph algorithms allows the conversion of data into modeled objects. We skip the description of this process since it is beyond the scope of this paper. Refer to Gimblett (2002) for an integration of GIS and agent-based models.

We used data obtained from the San Francisco Bay area shown in Fig 5, which includes 207 nodes and 338 arcs. In this example, the traffic network is represented by a bi-directional graph, unlike the previous example. We defined only one destination node at the center of San Francisco city, as indicated in Fig 5. A set of $Q_j(i, l)$ representing $\overline{F}_j$ is assigned to each node j in $\{1,2,\ldots,207\}$ including the destination node itself. Note that every element in the family of such sets specializes in the sole destination. Arcs are categorized into one of "limited access highways," "highways," and "others," and we defined travel time functions to each of them as $t_i = len_i/(3.0 - 0.1v_i)$, $t_i = len_i/(2.0 - 0.2v_i)$, and $t_i = len_i/(1.0 - 0.3v_i)$, respectively, where lengths are given by pixel unit. With the assumption that travel demand to this destination is uniformly distributed among all the nodes, two agents are instantiated at each time step at randomly chosen nodes except for the destination. This means that there are 207 O-D pairs with 207 origins. Though there are more than one origin, the number of destination node is one, implying that we only need to allocate one set of $Q_j(i, l)$ to each node j in J. It is assumed that tolls will not affect agents' decisions, thus only the estimated travel times to destination stored in each node as $Q_j(i, l)$ act as factors to drive agents. Initially, zero is assigned to $Q_j(i, l)$ for every i, j, and l in order to attract exploration of all the nodes for all possible states and strategies. Additionally, in order to tempt agents to head for the destination node, the value of $-10{,}000+R_j$, rather than mere R_j, is used upon the arrival at destination to update $Q_j(i, l)$ that the agent referred in the previous node.

During the first 500 time steps, agents' movements appear random rather than rational. We call this exploration period. We may employ the A* algorithm, for example, to shrink the duration of this period. Once the exploration period is over, agents acquire some habits of using particular paths that they have found to be optimal. For example, many of them rush on Bridge 2 (San Francisco-Oakland Bay) shown in right side of Fig 6. But this causes congestion on the bridge, turning all the paths which include this bridge to be less optimal. As a consequence, divergence occurs so that some agents currently using Bridge 2 are reallocated to Bridge 1 (Golden Gate) and Bridge 3 (San Mateo-Hayward). We call this the adjustment period. The state converges to traffic equilibrium after the adjustment period, as in the previous section. At this time, most, if not all, nodes store the sufficiently accurate estimated time, $Q_j(i, l)$, in equilibrium. We draw a convex hull of the nodes with $Q_j(i,l) \leq 100$ and another of nodes with $Q_j(i,l) \leq 200$, as shown in left side of Fig 6. Notice that the convex hull of $Q_j(i,l) \leq 100$ is longer to the south of destination than to the east and north. It can be inferred that the cause of this is the availability of more choices on land to the south than on the water body with fewer arcs (bridges). Additionally, we can observe the convex hull of $Q_j(i,l) \leq 200$ being strongly convex to the east. This is attributed to the existence of limited access highways leading to that region.

SUMMARY AND FURTHER APPLICATIONS

We have seen a strong relationship among the classical theory of traffic equilibrium, game theoretic approaches, and multi-agent reinforcement learning, particularly through the use of the Bellman equation. For the complex features and the dynamic nature of traffic network problems, game theoretic approach would be difficult with a standard static method. With the reinforcement method discussed in Sec 3 and Sec 4, however, equilibria and Nash equilibria can be identified (if any exists) by empirical and/or simulated data. We also verified that computational simulation of multi-agent reinforcement learning generate emergent equilibrium that agrees with the classical theory, in the sense that the travel times of all the used paths are equal to each other and to the minimum travel time, characterized by simulation results of very low variance among them. With these relationships, we find that these three paradigms are substitutable as well as complementary.

The attempt of Sec 4.2 is aimed at a real application. Though we need more precise and detailed data to apply it to the real traffic network, the basic principle we employed seems applicable. One possible application is to forecast the effects of governmental decisions such as addition of lanes (capacity improvement), closure of road segments, and temporary constructions. Since it takes some positive time for agents to adapt to the new equilibrium with the delayed response to the road supply functions, the approach presented in this paper reflects the reality better than the standard static approaches. Another possible application is the extension of intelligent transportation systems (ITS) based on $Q(i, l)$, where client systems on agents' vehicles and a central server communicate. Suppose client systems can exchange or share $Q(i, l)$ values through a server-side database, and the flow amounts v_i on each arc i can be monitored by server-side system. Then it is expected to be able to efficiently navigate agents with online optimization of decisions. As an instance of concrete technological example, Choy *et al* (2002) showed a method that uses fuzzy-neuro evolutionary hybrid techniques with online reinforcement learning to enable real time traffic control systems. This adds a technological application as well as socio-political applications of the model we have discussed.

ACKNOWLEDGMENT

We are grateful to the anonymous peer reviewers for their generous, to-the-point, and helpful comments. We also appreciate the comments given by the audience in the Third International Workshop of Computational Intelligence in Economics and Finance, 2003, NC, USA.

REFERENCES

Bandura, A. (1977) Social Learning Theory. General Learning Press, NY.

Beckman, M.J., McGuire, CB., & Winston, CB. (1956). Studies in the economics of transportation. Cowles Commission Monograph, Yale University Press, CN.

Bellman, R.E. (1957). Dynamic programming. Princeton University Press, NY.

Choy, M.C., Srinivasan, D., & Cheu, RL. (2002). Hybrid cooperative agents with online reinforcement learning for traffic control. The Proceedings of IEEE FUZZ 2002, Honolulu, HI USA. (2): 1015-1020.

Gimblett, R.H. (2002). Integrating geographic information systems and agent-based modeling techniques: for simulating social and ecological processes. Oxford University Press, NY.

Roth, A.E., & Erev, I. (1995). Learning in extensive-form games: experimental data and simple dynamic models in the intermediate term. Games and Economic Behavior. Special Issue: Nobel Symposium 8: 164-212.

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. Machine Learning 3, 9-44.

Sutton, R.S., and Barto, AG. (1998). Reinforcement learning: an introduction. The MIT Press, MA.

Taylor, P.D., & Jonker, L. (1978). Evolutionarily stable strategies and game dynamics. Mathematical Bioscience 40: 145-156.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. Communications of the ACM, 38 (3), 58-68.

Wardrop, J.G. (1952). Some theoretical aspects of road traffic research. Proceedings of the Institution of Civil engineers, Part II (1), 325-378.

Watkins, C.J.C.H. (1989). Learning from Delayed Rewards. Ph.D. thesis, Cambridge University.