Reinforcement learning: Introduction to theory and potential for transport applications[1]

Baher Abdulhai and Lina Kattan

Abstract: The aim of this paper is to develop insight into the potential of reinforcement learning (RL) agents and distributed reinforcement learning agents in the domain of transportation and traffic engineering and specifically in Intelligent Transport Systems (ITS). This paper provides a crystallized, comprehensive overview of the concept of RL and presents related successful applications in the field of traffic control and transportation engineering. It is divided into two parts: the first part provides a thorough overview of RL and its related methods and the second part reviews most recent applications of RL algorithms to the field of transportation engineering. Finally, it identifies many open research subjects in transportation in which the use of RL seems to be promising.

Key words: reinforcement learning, machine learning, traffic control, artificial intelligence, intelligent transportation systems.

Résumé : L’objectif visé par cet article est de développer un aperçu du potentiel des agents d’apprentissage par renforcement (« reinforcement learning : RL ») et des agents d’apprentissage par renforcement réparti dans le domaine de l’ingénierie des transports et de la circulation, plus spécifiquement dans les systèmes intelligents de transport (SIT). Cet article présente un survol clair et complet du concept de l’apprentissage par renforcement et montre des applications réussies dans le domaine du contrôle de la circulation et de l’ingénierie des transports. Il est divisé en deux sections : la première fournit un survol précis de l’apprentissage par renforcement et ses méthodes connexes, et la deuxième examine les plus récentes applications des algorithmes d’apprentissage par renforcement dans le domaine de l’ingénierie des transports. Finalement, l’article identifie plusieurs avenues de recherche possibles dans lesquelles l’apprentissage par renforcement pourrait être appliqué avec succès.

Mots clés : apprentissage par renforcement, apprentissage machine, contrôle de la circulation, intelligence artificielle, systèmes intelligents de transport.

[Traduit par la Rédaction]

Introduction

The very essence of intelligent transportation systems (ITS), arguably, is to treat transportation networks as real-time controllable systems. Consequently, continuous real-time monitoring, diagnosis, formulation, and dissemination of control all become essential elements. The “real-time” nature of the above feedback loop is often quite problematic because of the immense size of the system at hand. Conventional optimal control methods, dynamic programming for instance, suffer from the “curse of dimensionality”, whereas the large dimensionality of the system at hand and exponential growth of its possible states prohibit the attainment of an optimal solution in a practically short time interval even on the fastest computers available today. Potential solutions to this problem are continuously under investigation by researchers including an array of methods that settles for faster but suboptimal control of the whole system or possibly portions of it. One possible approach of interest to us from which this problem can be tackled is through the use of machine learning techniques from artificial intelligence (AI), particularly reinforcement learning (RL). Reinforcement Learning has two key advantages over conventional control methods: the potential of learning how to control a larger system in a short time and the ability to do so with or without a model of the system. Since transport system models are far from superb, the latter advantage is of key importance. As the rest of the AI field, the idea of machine learning via reinforcing favorable actions based on feedback from environment of an agent is not very new but rather has been around for decades. All efforts, however, were fairly fragmented and vaguely related (or unrelated), and none have been applied to control transportation systems. Over the last few years, however, the RL field matured significantly. A particular noteworthy effort is the outstanding work by Sutton and Barto (1998), which cleverly ties the pieces of the puzzling field together. Very recently as well, a handful of transportation researchers became aware of RL and its po-

Received 9 October 2002. Revision accepted 19 February 2003. Published on the NRC Research Press Web site at http://cjce.nrc.ca on 28 November 2003.

B. Abdulhai[2] and L. Kattan. Intelligent Transportation Systems Center and Testbed, Department of Civil Engineering, University of Toronto, Toronto, ON M5S 1A4, Canada.

Written discussion of this article is welcomed and will be received by the Editor until 30 April 2004.

[1]This article is one of a selection of papers published in this Special Issue on Innovations in Transportation Engineering.
[2]Corresponding author (e-mail: baher@ecf.utoronto.ca).

tential for transport applications and embarked on exploratory work, as will be detailed later in the paper. We think, however, that RL deserves a closer look by a larger community of transportation system researchers and practitioners, and hence we offer this paper. The main objective of this paper is to crystallize the main concepts of RL, survey the applications to transportation to date, and identify potential for future applications. This paper is intended to be an eye-opener for the transportation community at large, a starting point for those who develop interest in RL, and a potentially useful reference for mid-to-high range experts. Advanced references are also provided.

The paper is divided into two main sections: the first section reviews the concepts of RL with special effort to define the bigger picture while providing a satisfying level of detail, and the second section reviews most recent applications of RL algorithms to the field of transportation systems and identifies many open research subjects in transportation in which the use of RL seems to be promising.

Overview of reinforcement learning

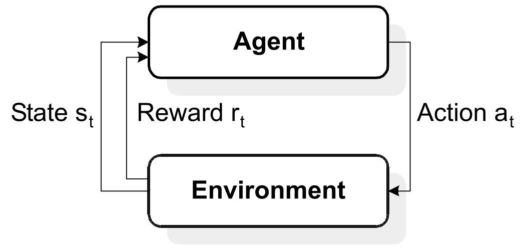
Reinforcement learning has adapted key ideas from various disciplines, namely, machine learning, operations research, control theory, psychology, and neuroscience, to produce some prominently successful engineering applications (Sutton and Barto 1998). RL is much more focused on goal-directed learning through interaction than other approaches to machine learning. This part of the paper covers keys characteristics and computational aspects of RL. It introduces the essential distinguishing features and elements of the RL problem, namely, the problem of learning policies, the concept of rewards, and the Markov decision process (MDP). It also focuses on the three classes of methods to solve the RL problem, namely, dynamic programming algorithm and its relation with MDP and Bellman equation, Monte Carlo methods, and temporal difference learning methods. More advanced RL methods that unify the basic ideas of the above three streams of methods are also described, such as eligibility traces, generalization, and the related role of neural networks in RL, and finally the integration of learning and planning methods in RL.

Reinforcement learning formulation

Reinforcement learning concerns the problem of a learning agent interacting with its environment to achieve a goal (Sutton 1999). Instead of being given explicit examples of desired behavior, the learning agent must discover by guided *trial and error* how to behave to get the most reward (or reinforcement signal). Reinforcement learning has become popular as an approach to artificial intelligence because of its simple algorithms and mathematical foundations (Sutton and Barto 1998; Bertsekas and Tsitsiklis 1996) and also because of a series of successful applications (Tesauro 1995; Crites and Barto 1996; Zhang W., and Dietterich 1996). In addition, the basic advantage of RL compared to other learning approaches is that it requires no information about the environment except for the reinforcement signal (Narendra and Thathachar 1989).

During learning the agent tries some actions (i.e., output values) on its environment; then it is reinforced by receiving

Fig. 1. Reinforcement learning framework (Sutton and Barto 1998).



a scalar signal (the reward or penalty) of its actions (Perez 1998). In RL, the reward signal passes from the environment to the agent. The goal of the agent is to maximize the total amount of reward it receives over time. Thus, RL algorithms selectively retain the outputs (actions) that maximize the reward received over time. This mapping from perceived state of the environment to perceived actions to be taken when in those states is called policy.

Reinforcement learning basic elements: The agent–environment interface

The learner or the decision-maker is called the agent, and everything it interacts with is called the environment (Fig. 1). The agent learns to perform an appropriate action by interacting with the environment: the agent selecting actions and the environment responding to those actions by presenting new situations and by passing a reward signal (evaluative feedback) to the agent.

Reinforcement learning tasks are generally treated in discrete time steps. At each time step t, the learning system receives some representation of the state of the environment s_t, it takes an action a_t, and one step later it receives a scalar reward r_t from the environment to indicate the desirability of this action. The environment then responds by presenting new situations to the agent (state s_{t+1}) (Fig. 1).

Trial and error search and delayed rewards

The two basic concepts behind RL are trial and error search and delayed rewards (Sutton and Barto 1998). Unlike most forms of machine learning, the agent is never told what the right actions are, but instead must discover which actions yield the most reward by trying them. Besides this, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. The task of the agent is to learn from this indirect, delayed reward, to choose a sequence of actions that produces the greatest cumulative reward (Mitchel 1997). The agent can acquire optimal strategies from delayed rewards even when the agent has no prior knowledge of the effects of its actions on the environment.

Exploration, exploitation, and action selection

Furthermore, one of the key aspects of RL is the trade-off between exploitation and exploration. To accumulate the most reward, the agent must prefer and (or) exploit the best experienced actions. However, it has to try new actions, i.e., explore, to discover better selections of action for the future (Sutton and Barto 1998). The dilemma is that neither exploitation nor exploration can be pursued exclusively without failing at the task. Explorative actions may even lead to penalties (negative rewards); however, the learning agent reinforces itself from these failure lessons. In the long term, these accumulated fiascos will lead to success rather than disaster (Jang et al. 1997). The ϵ-greedy and the softmax action selection algorithms are examples of methods used to balance exploration and exploitation in RL problems.

An ϵ-greedy learner behaves greedily most of the time but every once in a while, say with small probability ϵ, selects an action at random, uniformly, independently of the action-value estimates. This near greedy action selection rule is called ϵ-greedy method. The advantage of the ϵ-*greedy* method is that, as the number of trials increase, every action is sampled an infinite number of times. The probability of selecting the optimal action converges to greater than $1-\epsilon$ (Sutton and Barto 1998).

The disadvantage of the ϵ-greedy method is that it chooses equally among all actions, irrespective of the estimated reward value of each action. This means that it is just as likely to choose the worst appearing action, as it is to choose the next-to-best. To decrease the cost of failure in real time applications in tasks where the worst actions can be drastic, actions with higher estimated rewards can be chosen with greater priority than actions with fewer estimated rewards. Thus selection of action probabilities can be a graded function of estimated values. These are referred to as softmax action selection rules (Sutton and Barto 1998). The most common softmax methods use a Gibbs or Boltzman distribution and yield the following probability of action selection:

$$[1] \qquad P_{\mathrm{r}}(\text{action\_ } a \text{\_ is\_ selected}) = \frac{e^{Q_{\mathrm{r}}(a)\,\tau}}{\sum_{b=1}^{n} e^{Q_{\mathrm{r}}(b)/\tau}}$$

where $Q_{\mathrm{r}}(a)$ is action a value and τ is a temperature parameter for an annealing process. In other words, higher temperature causes all the actions to be nearly probable whereas low temperature causes greater difference in selection probability for actions that differ in their value estimates; in the limits as $\tau = 0$, action selection becomes greedy. When the cooling temperature is employed, exploration in the solution space will gradually converge to greediness.

Other reinforcement learning elements

In addition to the agent, the environment, and the reward, one can identify four main subelements to a RL system: policy, reward function, value function, and, possibly, model of the environment (Sutton and Barto1998).

A policy defines the way learning agent behaves at a given time. Roughly speaking, policy is a mapping from perceived states of the environment to actions to be taken in those states

A reward function defines the goal in a RL problem. It maps perceived states (or state–action pairs) of the environment to a single number, a reward, indicating the intrinsic desirability of the state.

While a reward function indicates what is good in an immediate sense, a value function specifies what is good in the long run. The single objective of reinforcement-learning agent is to maximize the total reward it receives in the long run. The value of a state is the total amount of reward an agent can expect to accumulate over the future starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states. For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards.

The final element of some RL systems is an optional model of the environment, which mimics the behavior of the environment. As such, given a state and action the model might predict the resultant next state and next reward. Such models are used for planning, meaning any way of deciding on a course of action by considering possible future situations before they are actually experienced. Models of the environment are optional, as the agent can directly learn from interaction with the real environment.

Markov chain

Modern reinforcement learning research uses the formal framework of Markov decision processes (MDPs) (Sutton and Barto 1998). An environment is said to satisfy the Markov property if its state signal completely summarizes the past sensations in such a way that all relevant information is retained. As mentioned previously, the agent and environment interact in a sequence of discrete time steps, $t = 0$, 1, 2, 3, etc. At each step, the agent perceives the environment to be in a state, s_t, and selects an action, a_t. In response, the environment makes a stochastic transition to a new state, s_{t+1}, and sends out a numerical reward, r_{t+1}. The dynamics can be defined by specifying the complete probability distribution for all a, r and all possible values of the past events: s_t, a_t, r_t, s_{t-1}, a_{t-1}, …, r_1, s_0, a_0.

$$[2] \qquad P^{a}_{ss'} = P_r\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\right\}$$

where $P^{a}_{ss'}$ defines the probability of being in a given state s' at time $t + 1$ depending on all past states and actions.

If the state signal has the Markov property, then the response of the environment at $t + 1$ depends only on the state and action representations at t (Ballard 1997), in which case the dynamics of environment can be defined only by specifying:

$$[3] \qquad P^{a}_{ss'} = P_r\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\right\} \qquad \text{for all } s', r, s_t, \text{ and } a_t$$

where $P^{a}_{ss'}$ defines the probability of being in a given state s' at time $t + 1$ depending only on the current state s at time t while taking action a.

A RL task satisfying the Markov property is called a Markov decision process (MDP). If the states and action space are finite then it is called finite Markov decision process (finite MDP).

Similarly, given any current state and action, s and a, together with any next state, s', the expected value of the next reward is:

$$[4] \qquad R^{a}_{ss'} = P_r\left\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\right\}$$

These two quantities, $P^{a}_{ss'}$ and, $R^{a}_{ss'}$, completely specify the most important aspects of the dynamics of a finite MDP.

Note that the above notations in MDP are for discrete states and rewards, but they can easily be extended to include continuous states and rewards (Sutton and Barto 1998).

Reinforcement learning basic solution approaches

The three methods described next, namely dynamic programming methods, Monte Carlo techniques, and temporal difference learning methods, are the three basic classes of methods for solving the RL problem and computing the optimal policy. Dynamic programming is a planning method and assumes the existence of a model of the environment and its transition probability, whereas the other two are learning methods and can learn solely from raw experience without using a model of the environment.

Dynamic programming methods

Reinforcement learning is also known as neuro-dynamic programming (Bertsekas and Tsitsiklis 1996). Thus, RL algorithms are related to dynamic programming (DP) algorithms frequently used to solve optimization problems (Mitchel 1997). Note that DP refers to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment (e.g., MDP). Classical DP algorithms are of limited utility in RL because of their assumption of a perfect model and great computational expense (Sutton and Barto 1998). However, they are still very important theoretically and are essential foundation of RL. Reinforcement learning attempts to achieve much the same as DP only with less computation and without assuming a perfect model of the environment.

The task of the agent is to learn a policy $\pi: S \rightarrow A$, for selecting its next action a_t based on the current observed states s_t; that is, $\pi(s_t) = a_t$. In DP as in RL problems we can define the value of being in a state s under policy π as the expected discounted return starting in that state and following policy π (Sutton and Barto 1998). The function $V^{\pi}(s)$ mapping all states to their values is called the state–value function for the policy π and is defined as

[5] $$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots\} \\ &= E_{\pi}\left\{\sum \gamma^k r_{t+k+1} \mid s_t = s\right\} \end{aligned}$$

where γ is a discount factor $0 \le \gamma \le 1$.

A fundamental property of the value functions is that they satisfy particular recursive relationships also known as Bellman equations

[6] $$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^K r_{t+k+2} \mid s_t = s\right\} \\ V^{\pi}(s) &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^K r_{t+k+2} \mid s_t = s\right\}\right] \\ V^{\pi}(s) &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V^{\pi}(s')\right] \end{aligned}$$

The method usually followed in DP and RL to evaluate $V^{\pi}(s)$ is that of iterative solution and successive approximation with initial approximation as 0 and a successive approximation of V_{π}. This algorithm is known as iterative policy evaluation. Each successive iteration is obtained using the following Bellman equation as an update rule:

[7] $$\begin{aligned} V_{k+1}(s) &= E_{\pi}\{r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s\} \\ V^{\pi}(s) &= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V_K(s')\right] \end{aligned}$$

Similarly, we denote $Q^{\pi}(s, a)$, the value of taking an action a in a state s under a policy π, as the expected return starting from s, taking action a, and thereafter following policy π:

[8] $$\begin{aligned} Q^{\pi}(s, a) &= E_{\pi}(R_t \mid s_t = s, a_t = a) \\ &= E_{\pi}\{\gamma^k r_{t+k+1} \mid s_t = s, a_t = a\} \end{aligned}$$

We call Q^{π} the action-value function for policy π.

In DP methods as well as in other RL methods, the value functions V^{π} and Q^{π} can be estimated by bootstrapping. This refers to updating the value of a state or the value of an action based on older estimates of the successor states of that state.

The state value (state–action value) function defines the desirability or utility of being in a given state (or state–action pair) when following policy π. Additionally, the value of states defines a natural ordering of policies (Sutton 1999). Policy π is better than or equal to policy π' if and only if $V^{\pi}(s) \le V^{\pi'}(s)$ for all states s. For finite MDPs there are always one or more policies that are better than or equal to all others. These are the optimal policies denoted by π^*. Thus, we define optimal state value function V^* and optimal action value function $Q^*(s, a)$ as

[9] $$\begin{aligned} V^*(s) &= \max_{\pi} V^{\pi}(s) && \text{for all } siS \\ Q^*(s,a) &= \max_{\pi} Q^{\pi}(s,a) && \text{for all } aiA \end{aligned}$$

The use of Q action value function allows the formulation of the policy improvement theorem (Ballard 1997), which is the core idea of both RL and DP technique. The policy improvement theorem indicates that incrementally changing the policy to the action that is locally best is also the right global thing to be done.

Monte Carlo techniques

The value functions V^{π} and Q^{π} can be estimated from experience. For instance, if an agent follows policy π and separate averages are kept for each action taken in a state, then these averages will converge to the action values, $Q^{\pi}(s, a)$. Estimation methods of this kind are called Monte Carlo methods because they involve averaging over many random samples of actual returns. Since a value of a state is the expected return, this average converges to a good approximation to the value function (Sutton and Barto 1998). Thus, the state value function and action value function are determined as

[10] $$\begin{aligned} V^{\pi}(s) &= \text{average (Returns}(s)) \\ Q^{\pi}(s,a) &= \text{average (Returns}(s,a)) \end{aligned}$$

where Returns(s) are the end of episode returns or sequence of rewards generated by beginning at states s and repeatedly using policy π to select actions. Similarly, Returns(s, a) are the end of episode returns or sequence of rewards generated by beginning at states s and taking action a and then repeatedly using policy π to select actions.

Monte Carlo methods differ from DP methods in two ways (Sutton and Barto 1998):

- First, they operate on sample experience. Therefore, they can be used for direct learning directly through interaction with the environment without needing a model of the environment dynamics.
- Second, they do not bootstrap; i.e., they do not build their value estimates for one state on the basis of the estimates of successor states.

However, MC methods are defined only for episodic tasks and learn only from the tail of an episode. In other words, it is only upon the completion of an episode that value estimates and policies are changed. Another complication of MC methods is that estimation of value of all the actions from each state is needed, not just the currently favored one to compare alternatives. One approach is to ignore this problem by assuming that episodes begin with state–action pairs randomly selected to cover all possibilities. Such exploring starts can sometimes be arranged in applications with simulated episodes but are unlikely in learning from real experience. Instead, one of two general approaches can be used:

- In *on-policy* methods, the agent commits to always exploring and tries to find the best policy that still explores.
- In *off-policy* methods, the agent also explores, but learns a deterministic optimal policy that may be unrelated to the policy followed.

Temporal difference learning methods

A central idea of reinforcement learning is called temporal difference (TD) learning. Temporal difference methods are general learning algorithms to make long-term predictions about dynamical systems (Perez 1998). Temporal difference methods are a class of incremental learning procedures specialized for prediction whereby credit is assigned based on the difference between temporally successive predictions (Sutton 1988). In fact, TD learning methods combine ideas from both DP and MC methods:

- Like MC methods, TD methods can learn directly from raw experience without a dynamic model of the environment; and
- Like DP, TD methods bootstrap: they update estimates based in part on other learned estimates, without waiting for a final outcome.

While MC methods wait till the end of an episode to determine the increment to $V(s_t)$ (only then is R_t known), TD-methods wait until the next time step: at time $t + 1$ they immediately form a target and update using the observed reward r_{t+1} and the estimate $V(s_{t+1})$ (Sutton and Barto 1998). For instance, the simplest form of TD known as TD(0) is:

[11] $$V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - \mathrm{V}(s_t)]$$

where α is a step-size parameter $0 \le \alpha \le 1$

Temporal difference methods have an advantage over DP methods in that they provide a way of finding an optimal policy purely from experience without requiring a model of the environment's dynamics, i.e., of its rewards and next-state probability distributions. They also have advantages over MC methods in that they are naturally implemented in an online, fully incremental fashion: while with MC methods one must wait until the end of an episode to find out the return known. In TD methods one needs only wait for one time step for the return.

Two types of TD algorithms are presented next, namely the on-policy SASRA algorithm and the off-policy Q-learning algorithm.

SARSA

The name SARSA comes from the fact that the updates are done based on the quintuple (s, a, r, s′, a′), namely, the current state, current action, resulting reward, next state, and next action. The SARSA algorithm is an online TD method that learns action–value functions by a bootstrapping mechanism, that is, by making estimations based on previous estimations. At every time step, SARSA updates the estimations of the action-value functions Q(s, a) using the quintuple (s, a, r, s′, a′). SARSA is an on-policy version of the well known Q-learning algorithm, where the learned action-value function Q directly approximates the optimal action-value function, denoted by Q*(s, a):

[12] $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

where α is a positive step-size parameter and γ is a discount factor. If s_{t+1} is terminal then:

[13] $$Q(s_{t+1}, a_{t+1}) = 0$$

SARSA is an on-policy algorithm in the sense that it is learning and improving the same policy that is followed in selecting the actions. The advantage of SARSA over other TD-algorithms lies in its high online performance.

Q-learning

The development of Q-learning (an off-policy TD control) is seen as one of the most important breakthroughs in RL. Its simplest form, 1-step Q-learning uses the experience of each state transition to update one element of a table (Sutton and Barto 1998). This table, denoted *Q*, has an entry, *Q*(*s*, *a*), for each pair of state, *s*, and action, *a*. Upon the transition *st, st* + 1, having taken action *at* and received reward *rt* + 1, this algorithm performs the following update:

[14] $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\, [r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

where α is a positive step-size parameter and γ is a discount factor. Decreasing the training rate over time is one of the conditions necessary for convergence of the algorithms to an optimum policy in stochastic environment.

The simplest 1-step Q-learning algorithm in procedural form is as follows:

```
  Initialize Q(s, a) arbitrary
  Repeat (for each step in the episode)
  Initialize s,
  Repeat (for step in the episode):
    Choose a from s using policy derived from
    Q(e.g., ϵ-greedy)
    Take action a, observe r and s′
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\, [r_{t+1} + \gamma\max_a Q(s_{t+1}, a_{t+1})$
    $- Q(s_t, a_t)]$;
    $s \leftarrow s'$;
Until s is terminal
```

Note that the 1-step Q-learning used to find the optimal policy for a Markov decision process (MDP) resembles online form of the dynamic programming value iteration method (Pendrith 2000); whereby in each learning step, is embedded a policy improvement step and a truncated policy evaluation step. Furthermore, Q-learning is an off-policy algorithm in the sense that it is gaining useful experience even while exploring actions that may later turn out to be non-optimal. This results in Q-learning having potentially poor online performance, especially during early learning, as compared to SARSA algorithm.

n-step temporal difference learning

Therefore, Monte Carlo methods perform a backup for each state based on the entire sequence of observed rewards from that state until the end of the episode. On the other hand, the one-step TD, is based on just the next reward. One kind of intermediate method, then, would perform a backup based on an intermediate number of rewards: more than one, but less than all of them until termination (Sutton and Barto 1998). For instance: a 2-step backup would be based on the first two rewards and the estimated value of the state two steps later. Similarly we could have 3-step backups, 4-step backups, etc.

This definition enables us to simply treat MC methods as the special case of infinite-step returns. Therefore, the *n*-step TD methods thus form a family of valid methods, with 1-step TD methods and Monte Carlo methods as extreme members:

$$\text{TD (1-step)} \leftarrow \text{TD}(n\text{-step}) \rightarrow \text{Monte Carlo}$$

Reinforcement learning more advanced solution approaches

When the above three elementary RL methods are combined they can produce a number of more powerful and efficient learning methods. The three ways of achieving this is through eligibility traces, function approximation and integrating planning and learning methods together as is explained in the following section.

Eligibility traces

Another common extension, eligibility traces, allows credit for a good state transition to spread more quickly to the states that preceded it, again resulting in faster learning (Sutton 1999). Almost any TD methods, e.g., Sarsa or Q-learning can be combined with eligibility traces to obtain a more general method able to learn more efficiently. Eligibility traces can be thought of as an exponential weighted adjustments to predictions occurring *n* steps in the past; more recent predictions make greater weight changes. This may match biological brain strategies for deciding how strongly recently received stimuli should be used in combination with current stimuli to determine actions (Jang et al. 1997). This type of TD learning is denoted TD(λ) algorithm, where the λ refers to the use of an eligibility trace (trace-decay parameter).

The TD(λ) algorithm can be understood as one particular way of averaging *n*-step backups (Sutton and Barto 1998). This average contains all the *n*-step backups, each weighted proportional to λ^{n-1}, where $0 \le \lambda \le 1$. In addition, a normalization factor of $1 - \lambda$ ensures that the weights sum to 1. The resulting backup is towards a return, called the λ-return, defined by:

$$[15] \quad R_t^{\gamma} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$$

If $\lambda = 0$, the overall backup reduces to its first component, the 1-step TD backup, whereas, if $\lambda = 1$, the overall backup reduces to just its last component, the Monte Carlo backup. Thus, when TD methods are augmented with eligibility traces they produce a family of methods spanning a spectrum that has Monte Carlo methods at one end and 1-step TD methods on the other.

Methods using eligibility traces require more computation than 1-step methods, but in return they offer significantly faster learning, particularly when rewards are delayed by many steps.

Reinforcement learning, generalization and function approximation

The use of tabular representation to store state or state–action values is questionable when RL is applied to more realistic environments that entail many possible states (Jung et al. 1997). In many tasks to which we would like to apply RL, most states encountered will never have been experienced exactly before. Furthermore, the state or action spaces may include continuous variables or very large number of states or even infinite state. One possibility is to generalize from previously experienced states to ones that have never been seen. Thus, more sophisticated RL methods implement *V* and *Q* values not as a table but as a trainable parameterized function such as artificial neural networks and statistical curve fitting, radial basis function, coarse coding, tile coding (CMAC), and Kanerva coding, etc. (Sutton and Barto 1998). This enables generalization among states, which can greatly reduce learning time and memory requirements.

For instance, an estimated value function can be represented as a weighed linear function of a set of board features $f_1, \ldots, f_n$:

$$[16] \quad V(i) = w_1 f_1(i) + w_2 f_2(i) +, \ldots, + w_n f_n(i)$$

Thus, instead of almost infinite states, the value function is characterized by *n* weights. This compression allows the learning agent to generalize from states it has visited to states it has not visited (Russel and Norvig 1995).

As an example, Crites and Barto (1996) describe a neural network reinforcement learning approach for an elevator-scheduling task.

Therefore, when combined with suitable function approximation methods, RL algorithms can provide computationally tractable ways to approximate the solutions of very large scale stochastic optimal control problems (Sutton and Barto 1998; Betsekas and Tsitsiklis 1996). However, all algorithms do not always carry from tabular to “function approximation”. In fact, there are many consequences of introducing these imperfect value estimations by function approximations such as the divergence of Q-estimates, and on-going research tries to minimize their impact on the quality of the solution (Artificial intelligence depot).

Reinforcement learning: Integrating learning and planning

In general, RL based solely on TD methods is a slow process. To learn a world model and practice with the model can be useful in speeding up the learning process.

As mentioned previously, RL is roughly classified into model-free learning and model-based learning (Jang et al. 1997). Both types of learning involve estimating the same value functions. In addition, they are identical algorithms operating on two different sources of experience: real experience and simulated experience. In fact, in the planning model case, a model of the environment is used to simulate extensive interaction between the environment and the agent. This simulated experience is then processed by reinforcement learning methods just as if it had actually occurred (Sutton 1999).

Model free learning methods can be used in conjunction with model-based methods. Sutton's Dyna architecture implemented one way of blending these two direct and indirect methods that retains many of the advantages of each approach. The use of internal model by Dyna is proved to dramatically speed up trial-and-error learning processes.

Reinforcement learning: Its applicability, its advantages over other techniques, and its limitations

The objective of RL is to learn how to construct a solution by using experience gathered from many example problems that are similar to the one whose solution is required. "*Note that the RL approach makes sense when we are interested in solving an entire class of similar problems rather than just a single problem; it is only within such a context that "learning from experience" can be helpful.*" (Bertsekas and Tsitsiklis 1996, pp. 55–56). Existing RL techniques are distinguished from other supervised learning techniques in the approach with which they perceive learning as well as in the type of applications in question. On the other hand, RL is simply an extension of classical DP; however, it greatly enlarges the scale of problems that can practically be solved. Despite these potentials, RL algorithms are still facing many challenges that are the topic of current RL research.

Reinforcement learning versus other computational approaches

Reinforcement learning combines the fields of supervised learning and DP to yield powerful machine learning systems. Reinforcement learning is distinguished from these computational approaches by its emphasis on learning by the individual from direct interaction with its environment, without relying on supervision (such as supervised learning) or on complete models of the environment (such as DP).

In supervised learning a "teacher" provides the learning system with a set of training examples in the form of input–output pairs (Crites and Barto 1998). Supervised learning algorithms are useful in a wide variety of problems involving patterns classification, continuous prediction, regression, and function approximation. Supervised learning is an important type of learning but is not adequate for learning from interaction. In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all situations in which the agent has to act (Sutton and Barto 1998). Reinforcement learning can be useful in these more difficult problems, where training information comes from a "critic" providing scalar evaluation of the output that was selected rather than specifying the best output. The RL agent must instead be able to learn from its own experience; it uses training information that evaluates the actions taken rather than instructs by giving correct actions (Crites and Barto 1998). In addition, by combining RL with supervised learning techniques such as neural networks, many researchers are optimistic that classes of problems previously unsolvable will finally be solved (Sutton and Barto 1998).

As compared to DP, RL attempts to achieve much the same as DP only with less computation and without always assuming a perfect model of the environment (prior knowledge of the state transition probabilities and reward structure of the environment, e.g., MDP). Unfortunately, because of these two requirements (perfect model of the environment and great computational expense), an optimal solution to the DP problem is impractical when the number of state set is large (Sutton and Barto 1998). Still, DP algorithms are better suited to handling very large state spaces than competitor operation research methods, such as direct search and linear programming. In large scale problems, there is the possibility of approximating the solution using RL. Thus, unlike traditional DP, RL can be used to improve performance online while the agent and environment interact. This online learning focuses computation onto the areas of state–space that is actually visited in real life, ignoring the areas that are not visited (Crites and Barto 1998). Thus, with this use of sampling strategy instead of a model and with possibly function approximation, RL can apply to systems that are too large or complicated to explicitly enumerate the next-state transition probabilities, such as DP (Sutton 1999).

Reinforcement learning challenges

On the other hand, there are two main challenges in reinforcement learning research: (*a*) scaling up to large problems and (*b*) handling partially observable Markov decision problems (where the agent cannot sense the entire state of the environment). Firstly, it is often too memory expensive to store values of each state, since the problems can be quite complex. Solving this involves looking into value approximation techniques, such as decision trees or neural networks. There are many consequences of introducing these imperfect value estimations such as the divergence of Q-estimates in bootstrapping methods (Artificial intelligence depot). Secondly, because of limited perception of the state, it is often impossible to fully determine the current state. In fact, RL fails in problems where the learning agent cannot sense the state of its environment; therefore, the system is far from Markov and there is little or no advantage provided by addressing the state information. In such cases evolutionary algorithms may perform better, while the performance of the RL algorithms is often affected. Much research remains to be done in this respect. Some of the proposed approaches treat the problem as a partially observed MDP (POMDP); however, this approach is still computationally expensive.

Thus far, RL continues to be one of the most active areas of machine learning research today. One of the larger trends that RL is pursuing is towards greater contact between artificial intelligence and other engineering disciplines (Sutton and Barto 1998). The next part of this paper reviews such RL applications in the case of transportation engineering.

Reinforcement learning applications in transportation

Transportation systems optimization and control, which to date has largely relied on traditional operations research techniques like classical dynamic programming and linear and nonlinear programming problems, offers a rich, challenging set of problems for which both single and multi-agent reinforcement learning methods might be useful. Reinforcement learning appears to offer promising results in application to transportation processes where real-time, adaptive control is the key to improving effectiveness and efficiency (Abdulhai et al. 2001). The key advantages that reinforcement learning is able to offer in this regard are related to the ability of a control agent to directly learn the relationships between control actions and their effects on the environment while pursuing a goal, without the need for an explicit model of the environment. This part of the paper provides a survey review of such RL applications in the transportation field. Most of the applications found are related to the field of traffic engineering, specifically to traffic signal control. To the best of our knowledge, only one RL application is found in other transportation applications, specifically the domain of infrastructure maintenance.

Reinforcement learning application to traffic signal control and traffic management

From the machine learning community perspective, the traffic domain has much to it recommended as a fertile source of research problems for reinforcement learning or (multi-agent reinforcement learning). Clearly, each agent (a driver, the vehicle, or the traffic light) has independent control of its actions, but its behavior must take into account physical constraints as well as the behavior of many other agents. Another advantage is that there exist clear criteria for evaluation, such as maximizing traffic flow and minimizing lane changes. In addition, the domain supports complex behaviors of the overall system, such as traffic jams, even though the individual agents are relatively simple. Also, most artificial intelligence researchers have personal experience in traffic environments and, presumably, have good intuition about reasonable strategies and behaviors (Moriarty et al. 1998).

Recent research literature includes several related efforts where reinforcement-learning algorithms are applied to the problem of traffic control. For instance, Thorpe (1997) applied the SARSA RL algorithm to a simulated traffic light control problem using illegibility traces and greedy action selection. The goal of the RL agent is to minimize the time required to discharge a fixed volume of traffic through a road network. The performance of the SARSA is analyzed with four different representations of the current state of traffic, namely, the vehicle counts, the fixed distance representation, the variable distance representation, and the count or duration. In addition, four performance measures were calculated: the total number of simulation steps required for all vehicles to reach their destinations (the final goal state), the average vehicle travel time, the total number of stops made by all vehicles, and the average vehicle wait time. The tests conducted with the above four different state representations show that choice of state representation is critical to the success of the SARSA strategy. SARSA fixed-distance and variable-distance representations are most effective in reducing average vehicle wait time. However, Thorpe's approach does not appear to be oriented to real-time traffic signal control.

Bingham (2001) applied RL in the context of a neuro-fuzzy approach to traffic signal control. The aim of Bingham's work was to create an adjustable fuzzy traffic signal controller capable of modifying its own parameters in different traffic situations. A fuzzy traffic signal controller uses linguistic rules, such as "if the approaching traffic volume is large and the queuing traffic volume is small, then the green signal is long". The fuzzy concepts large, small, and long are presented using membership functions. In the studied neuro-fuzzy controller, the parameters of the fuzzy membership functions are adjusted using a neural network based on RL. The objective of the signal controller is to minimize the delay of vehicles. Depending on the traffic situation, the green phase can be extended with one or several seconds, and the output of the fuzzy controller (RL action) is EXT (green time extension (in seconds)) with linguistic values of zero, short, medium, and long. The role of the RL agent is to evaluate actions of the fuzzy controller; thus if the action had good consequences, the tendency to produce that action is strengthened (reinforced). The adopted traffic simulation environment is a two-phase controlled intersection to two-lane streets. The controller performance is measured by the delay of vehicles. The results of the simulation experience show that the incorporation of the RL algorithm in the fuzzy traffic signal control decreases the vehicular delay at constant traffic volumes with significant delay reduction at low traffic volumes. However, the learning algorithm is not found to be successful in situations with highly changing traffic volume. Martin and Brauer (2000) also presented a fuzzy model-based RL approach, fuzzy prioritized sweeping (F-PS), and applied it to the problem of optimal framework signal plan selection.

Abdulhai et al. (2001) applied a simple Q-learning technique to an isolated traffic signal in a two-phase-signal two-dimensional road network. Individual Poisson processes are used to generate vehicle arrivals on each approach to the system. Traffic movement within the system is simulated at a microscopic level with each road link divided into blocks. The state information available to the agent includes the queue lengths on the four approaches with the elapsed phase time. The isolated signal agent is operated with a fixed cycle length. Each second, the agent selects an action: either to remain with the current signal indication or change it. The reward (a penalty in this case) is the total delay incurred among successive decision points by vehicles in the queues of the four approaches. The objective of the agent in seeking the optimum policy is to maximize the accumulated reward (or minimize the accumulated penalty) over time. Balancing exploitation and exploration is implemented using either an ϵ-greedy policy (with $\epsilon = 0.9$) or a softmax action selection. In the case of the isolated intersection, two CMACs are used, one for the change action and one for the don't-change action. Three different traffic profiles (uniform traffic flows, constant-ratio traffic flows, and variable traffic flow) are tested to evaluate the performance of the Q-learning agent under varying conditions. The Q-learning agent proves to be able to outperform the pretimed controller under these three conditions because of its ability to adapt to minor random fluctuations in flow. Subsequent comparisons with other commonly used automated signal system control such as SCOOT are to be conducted in future stages of this research.

Wiering (2000) studied the use of multi-agent RL algorithms for learning traffic light controllers. The goal of the agents is to minimize the average waiting time of cars in a city. Reinforcement learning is used to learn value function estimating expected waiting times (EWTs) of cars until they have arrived at their destination address given different settings of traffic lights. Selected settings of traffic lights are based on different combination of the predicted waiting times of all cars involved. Weiring also examined how to use RL systems and the corresponding learned value functions to optimize the paths that cars take to arrive to their destination (optimal routes to destination). Such type of learning strategy is termed co-learning and can be used if all agents share the goal of using different policies to minimize the same value function. The interesting concern about co-learning is that different types of agents (cars and traffic lights in this case) use the same value functions to optimize their policies and in doing so they cooperatively try to minimize the same value function. As a test, Wiering (2000) used a network consisting of six nodes modeled in a grid with the objective to synchronously learning controllers for all nodes. The traffic simulator used consists of a network of two-lane roads. Cars that turn left at the next intersection use the left lane and cars that go straight ahead or turn right use the right lane; each lane is assumed to have its own traffic light. There is a single controller at each traffic node for setting all traffic lights. Cars, which cross an intersection, use their driving policy to select a new lane. New cars with a particular destination address are generated according to a random process to enter the simulated network. As such, a car is represented by its current traffic light, place, and destination address. The *Q*-function Q(tl, p, dest, action) denotes the EWTs of a car given the action of the traffic light and is stored in a lookup table. For learning the EWT of a car, Weiring used model-based RL where transition function is estimated by tracking movements of cars. The reward function used is simple: the cost for standing is 1, and 0 otherwise. After each time step, the transition function is updated and real-time dynamic programming is used to compute the value functions. Three different RL systems were designed: TC-1, which does not use communication among traffic lights, TC-2, which uses communication to make the transition function of the first car dependent on the number of cars standing at the next light, and TC-3, which uses this information to compute transition probabilities for all cars. Finally, Wiering (2000) used the same computed Q-functions to learn driving policies for cars, which allow for “co-learning” where both traffic light controllers and driving policies are optimized. The experimental results show that the RL algorithms can outperform nonadaptable traffic light controllers and that, in addition, optimizing driving policies is very useful. In fact, learning driving policies at the same time as learning traffic light controllers shows interesting co-learning phenomena: traffic nodes, which are quite busy and thus have a hard task minimizing overall waiting time are relieved by intelligent driving policies (i.e., cars are reactively spreading in the city) avoiding such intersections. Therefore, the use of communicated information (where global information is communicated between traffic light agents) is shown to help the RL systems to further optimize traffic light controllers. When the network starts to saturate the RL systems clearly outperform fixed controllers and also benefit significantly from co-learning driving policies. However, Weiring (2000) results were compared only with nonadaptable traffic controllers and no attempt was made to compare these results with advanced real time controllers.

Another example is the work carried out at the Daimler–Chrysler Research and Technology Center, where the research effort focuses mainly on the role of the vehicles themselves and their coordination in controlling traffic flow rather than the traditional traffic signals or ramp meters control (Langely). This novel approach to traffic management relies on a distributed scheme in which vehicles themselves select lanes, speeds, and routes. A simulated highway network populated by thousands of vehicles provides the research testbed. An RL system acquires control strategies based on experience with the traffic simulator, and a separate module learns to predict the expected loads along alternative routes. Therefore, the learning system consists of three components with (symbiotic, adaptive neuro-evolution) SANE RL as the backbone of the learning (proposed by Moriarty and Mikkulainen (1996)). The two other components are, respectively, a supervised learning from preexisting domain knowledge and a local learning strategy similar in spirit to temporal difference methods. The reward function involves not only individual vehicles but also overall traffic behavior, as measured by the average squared difference between actual and desired speeds of the vehicles. A number of related articles have been published, namely Moriarty and Langley (1998*a*), Moriarty et al. (1998), Moriarty and Langley (1998*b*), and Pendrith (2000). The results of these experimental studies have demonstrated that the strategies learned by this approach can achieve higher traffic throughput, maximize desired speeds in allowing drivers to match more closely their desired speeds, while reducing the number of lane changes (therefore reducing also accident rate). Intelligent lane selection is also shown to be robust in the presence of selfish drivers, and traffic performance is proved to improve even when as few as 5% percent of the cars cooperate (Pendrith 2000; Moriarty and Langley 1998*a*). Moreover, the results also indicate that the learned behaviors generalize to different traffic densities, different numbers of lanes, situations involving blocked lanes, and even in the presence of selfish drivers (Moriarty et al. 1998).

Reinforcement learning application in infrastructure management

Durango and Madanat (2001) proposed a TD learning RL framework to address the problem of developing maintenance and repair policies for facilities in transportation infrastructure management. Using a TD RL technique, Durango and Madanat (2001) were able to perform policy evaluation and policy selection without needing a model of the deterioration process of the facility. They applied this TD method to a pavement management simulation study and showed that TD methods can successfully fine-tune incorrect maintenance and repair policies. In the simulation study, eight states are used to represent pavement conditions rating. The RL agent (planning agency) has the choice among seven maintenance and repair actions available. The goal of the agent is to weigh the trade-off between reduced operating costs and increased maintenance and repair costs. Policies are evaluated by mapping the effect of the actions that are applied to the facility with the actual costs that are incurred. The performance measures of three different TD algorithms are tested, namely, SARSA, Q-learning, and TD(λ = 1). In all of these, TD methods, an ϵ-greedy action selection (ϵ = 0.1) policy is used. In addition, a nonlinear function approximation is applied to represent state action value function. The results show that the advantage of this approach lies mainly in its capability to adapt and fine-tune policies in situations where complete and correct models of facility deterioration are not (yet) available. However, the convergence rate for finding the optimal policy is shown to be slow; this is probably because of the limited number of observations used to update the value function.

Concluding remarks

This paper provided a comprehensive overview of RL and its related techniques. A number of RL applications in the transportation field were also reviewed. The incorporation of RL into the domain of transportation systems is relatively new but appears to be a promising area of research. The use of RL in optimizing traffic signal control is the most researched part. The likely application of RL in further areas of the transportation field needs to be explored and examined further. The ability or RL agent to exert real-time, adaptive control over a transportation process is potentially useful for a variety of transportation and specifically ITS services, including control of a system of traffic signals, control of the dispatching of paratransit vehicles, and control of the changeable message displays or other cues in a dynamic route guidance system (ATIS and ATMS) (Abdulhai et al. 2000). Other vital ITS application domains concern the possibility of applying RL for updating and estimating dynamic OD for a traffic area as well as to its inverse problem, i.e., normative dynamic traffic assignment (DTA), whereby we try to simultaneously optimize controls and routing. Most existing DTA models take account of representing control, which is considerably simpler than actually optimizing them. No existing DTA model is able to optimize controls and routing concurrently (Peeta 2001). RL when used to its full potential appears to offer promising results in this regard:

- For instance it is possible to decompose the DTA problem into several independent subproblems that can be solved by a team of learning agents (Multiagent RL). As such, every RL agent work on simpler subproblems (some agents learning control strategies (separate agent for each traffic signal, ramp metering, changeable message signs, etc., as well as other agents working on learning routing strategy) but all the RL agents work together to achieve a global goal such as minimizing total vehicular delay on the network, increasing throughput and increase safety.
- Another important feature of RL in DTA is its ability to use a traffic simulator to learn the optimal policies and then to resume on-going learning and adaptation when deployed in field and interacting with real experience. Furthermore, RL has the ability to use real experience to calibrate and find the deficiencies in the traffic model, which is still used as a laboratory for conducting further planning.

As such, this integrated dynamic traffic management and control system can adapt and respond to traffic conditions in real-time and maintain its integrity and stability within the total system.

References

Artificial intelligence depot [online]. Available from http://reinforcementlearning.ai-depot.com/ [accesssed on December 2001].

Abdulhai, B., Pringle, R., and Karakoulas, G.J. 2001. Reinforcement learning for ITS: Introduction and a case study on adaptive traffic signal control transportation research board, Transportation Research Board 80th Annual Meeting, Washington, D.C., 7–11 January 2001.

Abdulhai, B., Pringle, R., and Karakoulas, G.J. 2003. Reinforcement learning for true adaptive traffic signal control. ASCE Journal of Transportation Engineering. **129**(3): 278–285.

Ballard, D. 1997. An introduction to natural computation. The MIT Press, Cambridge, Mass.

Baird, L. 1995. Residual algorithms reinforcement learning with function approximation. *In* Proceedings of the 12th International Conference on Machine Learning, San Francisco, Calif., 9–12 July 1995. Morgan Kaufman Publishers. San Francisco, Calif. pp. 30–37.

Bertsekas, D.P., and Tsitsiklis, J.N. 1996. Neuro-dynamic programming. Athena Scientific, Belmont, Mass.

Bingham, E. 2001. Reinforcement learning in neurofuzzy traffic signal control. European Journal of Operation Research, **131**: 232–241.

Crites, R.H., and Barto, A.G. 1996. **9**. Improving elevator performance using reinforcement learning. *In* Advances in neural information processing systems. MIT Press, Cambridge, Mass. pp. 1017–1023.

Durango, P., and Madanat, M. 2001. Reinforcement learning models for transportation infrastructure management, Proceedings of the Second Berkeley–Tottori Joint Seminar on Evolution Processes of Transportation Systems: Analysis and Policy Implementation.

Jang, J., Sun, C., and Mizutani, E. 1997. Neuro-fuzzy and soft computing. Prentice Hall.

Langley, P. [online]. Available from http://newatlantis.isle.org/~langley/ [accessed on December 2001].

Mitchel, T. 1997. Machine learning, McGraw-Hill.

Martin, A., and Brauer, W. 2000. Fuzzy model-based reinforcement learning, European Symposium on Intelligent Techniques (ESIT), Aachen, Germany, 14–15 September 2000. pp. 14–15.

Moriarty, D., and Langley, P. 1998*a*. Learning cooperative lane selection strategies for highways. *In* Proceedings of the Fifteenth National Conference on Artificial Intelligence, Menlo Park, Calif., AAAI Press, pp. 684–691.

Moriarty, D., and Langley, P. 1998*b*. Distributed learning of lane-selection strategies for traffic management technical report. 98–2. Daimler-Benz Research and Technology Center, Palo Alto, Calif.

Moriarty, D.E., Handley, S., and Langley, P. 1998. Learning distributed strategies for traffic control. *In* Proceedings of the Fifth International Conference of the Society for Adaptive Behavior Zurich, Switzerland. pp. 437–446.

Narendra, K., and Thathachar, M. 1989. Learning Automata an introduction. Prentice-Hall.

Peeta. 2001. Foundations of dynamic traffic assignment: the past, the present and the future. Networks and Spatial Economics, **1**: 223–265.

Pendrith, M.D. 2000. Distributed reinforcement learning for a traffic engineering application *In* Proceedings of the 4th International Conference on Autonomous Agents, Barcelona, Spain. ACM, New York, N.Y., pp. 404–411.

Perez, A. 1998. Introduction to reinforcement learning [online]. Available from http://lslwww.epfl.ch/~aperez/RL/RL.html [accessed on December 2001].

Russel, S., and Norvig, P. 1995. Artificial intelligence: a modern approach. Prentice Hall Series in Artificial Intelligence, Englewood Cliffs, N.J.

Sutton, R. 1988. Learning to predict by the methods of temporal differences. Machine Learning, **3**: 9–44.

Sutton, R. 1999. Reinforcement learning: past, present and future? [online]. Available from http://www-anw.cs.umass.edu/~rich/Talks/SEAL98/SEAL98.html [accessed on December 2001].

Sutton, R., and Barto, A. 1998. Reinforcement learning: an introduction. MIT Press, Cambridge Mass.

Tesauro, G.J. 1995. Temporal difference learning and TD-Gammon. Communications of the ACM, **38**: 58–68.

Thorpe, T.L. 1997. Vehicle traffic light control using SARSA. Master's Project Report. Computer Science Department, Colorado State University, Colo.

Wiering, M.A. 2000. Learning to control traffic lights with multi-agent reinforcement learning, First World Congress of the Game Theory Society Games 2000, Utrecht, Netherlands, Basque Country University and Foundation B.B.V. Bilbao, Spain.

Zhang, W., and Dietterich, T. 1996. High-performance job-shop scheduling with a time-delay TDλ network. Advances in neural information processing systems, **8**: 1024–1030.

This article has been cited by:

1. Bekir Bartin. 2019. Use of learning classifier systems in microscopic toll plaza simulation models. *IET Intelligent Transport Systems* **13**:5, 860-869. [Crossref]
2. Chia-Hao Wan, Ming-Chorng Hwang. 2018. Value-based deep reinforcement learning for adaptive isolated intersection signal control. *IET Intelligent Transport Systems* **12**:9, 1005-1010. [Crossref]
3. Mohammad Aslani, Stefan Seipel, Mohammad Saadi Mesgari, Marco Wiering. 2018. Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown Tehran. *Advanced Engineering Informatics* **38**, 639-655. [Crossref]
4. Hyunjeong Jeon, Jincheol Lee, Keemin Sohn. 2018. Artificial intelligence for traffic signal control based solely on video images. *Journal of Intelligent Transportation Systems* **22**:5, 433-445. [Crossref]
5. Yizhe Wang, Xiaoguang Yang, Hailun Liang, Yangdong Liu. 2018. A Review of the Self-Adaptive Traffic Signal Control System Based on Future Traffic Environment. *Journal of Advanced Transportation* **2018**, 1-12. [Crossref]
6. Yiiun Cheng, Jun Peng, Xin Gu, Xiaovonz Zhang, Weirong Liu, Yinaze Yang, Zhiwu Huang. RLCP: A Reinforcement Learning Method for Health Stage Division Using Change Points 1-6. [Crossref]
7. H. M. Abdul Aziz, Feng Zhu, Satish V. Ukkusuri. 2018. Learning-based traffic signal control algorithms with neighborhood information sharing: An application for sustainable mobility. *Journal of Intelligent Transportation Systems* **22**:1, 40-52. [Crossref]
8. Antonella Ferrara, Simona Sacone, Silvia Siri. Emerging Freeway Traffic Control Strategies 293-311. [Crossref]
9. Mehmet Erkan Yuksel. 2018. Agent-based evacuation modeling with multiple exits using NeuroEvolution of Augmenting Topologies. *Advanced Engineering Informatics* **35**, 30-55. [Crossref]
10. Mohammad Aslani, Mohammad Saadi Mesgari, Marco Wiering. 2017. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C: Emerging Technologies* **85**, 732-752. [Crossref]
11. G. Bas, K. De Boo, A.M. Vaes-Van de Hulsbeek, I. Nikolic. 2017. MarPEM: An agent based model to explore the effects of policy instruments on the transition of the maritime fuel system away from HFO. *Transportation Research Part D: Transport and Environment* **55**, 162-174. [Crossref]
12. Bekir Oğuz BARTIN. 2017. ARAÇLARIN KRİTİK ARALIK KABUL KARARLARININ PEKİŞTİRMELİ ÖĞRENMEYLE SİMÜLASYONU. *Uludağ University Journal of The Faculty of Engineering* **22**:2, 161-178. [Crossref]
13. Cenk Ozan, Ozgur Baskan, Soner Haldenbilen. 2017. A Novel Approach Based on Reinforcement Learning for Finding Global Optimum. *Open Journal of Optimization* **06**:02, 65-84. [Crossref]
14. Saad Touhbi, Mohamed Ait Babram, Tri Nguyen-Huu, Nicolas Marilleau, Moulay L. Hbid, Christophe Cambier, Serge Stinckwich. 2017. Adaptive Traffic Signal Control : Exploring Reward Definition For Reinforcement Learning. *Procedia Computer Science* **109**, 513-520. [Crossref]
15. Patrick Mannion, Jim Duggan, Enda Howley. An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control 47-66. [Crossref]
16. Hossam Abdelgawad, Baher Abdulhai, Samah El-Tantawy, Alireza Hadayeghi, Brue Zvaniga. 2015. Assessment of self-learning adaptive traffic signal control on congested urban areas: independent versus coordinated perspectives. *Canadian Journal of Civil Engineering* **42**:6, 353-366. [Abstract] [Full Text] [PDF] [PDF Plus]
17. Cenk Ozan, Ozgur Baskan, Soner Haldenbilen, Halim Ceylan. 2015. A modified reinforcement learning algorithm for solving coordinated signalized networks. *Transportation Research Part C: Emerging Technologies* **54**, 40-55. [Crossref]
18. ANDRÉ DE PALMA, NATHALIE PICARD, MATTHIEU DE LAPPARENT. 2014. Risky Time Prospects and Travel Demand. *Mathematical Population Studies* **21**:4, 185-188. [Crossref]
19. Samah El-Tantawy, Baher Abdulhai, Hossam Abdelgawad. 2014. Design of Reinforcement Learning Parameters for Seamless Application of Adaptive Traffic Signal Control. *Journal of Intelligent Transportation Systems* **18**:3, 227-245. [Crossref]
20. Schmidt Shilukobo Chintu, Richard Anthony, Maryam Roshanaei, Constantinos Ierotheou. 2014. Intelligent and Predictive Vehicular Networks. *Intelligent Control and Automation* **05**:02, 60-71. [Crossref]
21. Samah El-Tantawy, Baher Abdulhai, Hossam Abdelgawad. 2013. Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and Large-Scale Application on Downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems* **14**:3, 1140-1150. [Crossref]
22. Kasra Rezaee, Baher Abdulhai, Hossam Abdelgawad. 2013. Self-Learning Adaptive Ramp Metering. *Transportation Research Record: Journal of the Transportation Research Board* **2396**:1, 10-18. [Crossref]

23. Pitipong Chanloha, Wipawee Usaha, Jatuporn Chinrungrueng, Chaodit Aswakul. Performance Comparison between Queueing Theoretical Optimality and Q-Learning Approach for Intersection Traffic Signal Control 172-177. [Crossref]

24. Samah El-Tantawy, Baher Abdulhai. Multi-Agent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC) 319-326. [Crossref]

25. Kasra Rezaee, Baher Abdulhai, Hossam Abdelgawad. Application of reinforcement learning with continuous state space to ramp metering in real-world conditions 1590-1595. [Crossref]

26. Ahmed Idris, Amer Shalaby, Khandker Habib. 2012. Towards a learning-based mode shift model: a conceptual framework. *Transportation Letters* **4**:1, 15-27. [Crossref]

27. Samah El-Tantawy, Baher Abdulhai. An agent-based learning towards decentralized and coordinated traffic signal control 665-670. [Crossref]

28. Zhaohui Yang, Kaige Wen. Multi-objective optimization of freeway traffic flow via a fuzzy reinforcement learning method V5-530-V5-534. [Crossref]

29. Samah El-Tantawy, Baher Abdulhai. 2010. Towards multi-agent reinforcement learning for integrated network of optimal traffic controllers (MARLIN-OTC). *Transportation Letters* **2**:2, 89-110. [Crossref]

30. Kaige Wen, Wugang Yang, Shiru Qu. A stochastic adaptive traffic signal control model based on fuzzy reinforcement learning 467-471. [Crossref]

31. Kaige W, Wugang Yang, Shiru Qu. Efficiency and equity based freeway traffic network flow control 453-456. [Crossref]

32. Zhipeng Shen, Chen Guo, Ning Zhang. 2010. A general fuzzified CMAC based reinforcement learning control for ship steering using recursive least-squares algorithm. *Neurocomputing* **73**:4-6, 700-706. [Crossref]

33. Kaige Wen, Wugang Yang, Shiru Qu. Freeway network traffic management based on distributed reinforcement learning 684-687. [Crossref]

34. Add Sadek, Nagi Basha. Self-Learning Intelligent Agents for Dynamic Traffic Routing on Transportation Networks 503-510. [Crossref]

35. Kaige Wen, Shiru Qu, Yumei Zhang. A Machine Learning Method for Dynamic Traffic Control and Guidance on Freeway Networks 67-71. [Crossref]

36. Bernard Adam, Ian F. Smith. 2008. Reinforcement Learning for Structural Control. *Journal of Computing in Civil Engineering* **22**:2, 133-139. [Crossref]

37. Kaige Wen, Shiru Qu, Yumei Zhang. A stochastic adaptive control model for isolated intersections 2256-2260. [Crossref]

38. Jing Xu, Wen-Sheng Yu, Fei-Yue Wang. Ramp metering based on adaptive critic designs 1531-1536. [Crossref]