This is a pre-publication draft of the paper, which was accepted for publication in Engineering Applications of Artificial Intelligence on 20/02/2012

An automated signalized junction controller that learns strategies by temporal difference reinforcement learning

Simon Box and Ben Waterson

Transportation Research Group, Faculty of Engineering and the Environment, University of Southampton, UK, SO17 1BJ.

Abstract

This paper shows how temporal difference learning can be used to build a signalized junction controller that will learn its own strategies though experience. Simulation tests detailed here show that the learned strategies can have high performance. This work builds upon previous work where a neural network based junction controller that can learn strategies from a human expert was developed (Box and Waterson, 2012). In the simulations presented, vehicles are assumed to be broadcasting their position over WiFi giving the junction controller rich information. The vehicle's position data are pre-processed to describe a simplified *state*. The state-space is classified into regions associated with junction control decisions using a neural network. This classification is the *strategy* and is parametrized by the weights of the neural network. The weights can be learned either through supervised learning with a human trainer or reinforcement learning by temporal difference (TD). Tests on a model of an isolated T junction show an average delay of 14.12 *s* and 14.36 *s* respectively for the human trained and TD trained networks. Tests on a model of a pair of closely spaced junctions show 17.44 *s* and 20.82 *s* respectively. Both methods of training produced strategies that were approximately equivalent in their equitable treatment of vehicles, defined here as the variance over the journey time distributions.

Keywords: , Intelligent Transportation Systems, signal control, machine learning, neural network, reinforcement learning, temporal difference, traffic, control, junction

1. Introduction

1.1. Background

Urban signalized road junctions are usually controlled by active systems (e.g. Vincent and Peirce (1988); Hunt et al. (1982)), which use sensors to measure

the state on the road. The state is then used by the control algorithm to inform decisions on which colour to set the traffic lights. Sensors such as inductive loops (Sreedevi, 2005) and microwave emitter/detectors (Wood et al., 2006) are commonplace and widely deployed in developed areas. New sensing technologies such as vehicle to infrastructure WiFi communications have been extensively investigated in recent years (Kompfner, 2008; COOPERS, 2010; SAFESPOT, 2010) leading to a Europe wide reservation of frequencies (IEEE 802.11p) for this type of communication.

The profusion of sensing technology leads to rich data that can be used for Urban Traffic Control (UTC). This enables the development of increasingly sophisticated control systems for signalized road junctions. In particular, data hungry *machine learning* algorithms can be employed to develop junction control systems that can learn improved strategies through various forms of training.

Recent important work on the optimisation of traffic signals has investigated a number of approaches including dynamic programming (Heydecker et al., 2007; Heung et al., 2005), genetic algorithms (Mikami and Kakazu, 1993), fuzzy-neural networks (Choy et al., 2003) and reinforcement learning (Chen and Heydecker, 2009). This work has shown how to use learning techniques to optimise parameters in signal control strategy or to select pre-defined strategies.

Here we are concerned with a pattern recognition approach where control decisions are made purely based on a classification of state space. Earlier work using this approach has shown how to use supervised learning to enable a junction controller to learn strategies from a human expert trainer (Box and Waterson, 2012). In this paper the approach is extended by the application of reinforcement learning to enable a junction controller to learn strategies through experience.

1.2. Context and motivation

Earlier work by the authors investigating the use of (vehicle transmitted) GPS + WiFi data in signal control has employed simulation to develop and evaluate control systems.

Under the *auctioning agent* control system (Waterson and Box, 2012) the road network is discretized into regions and software agents monitoring each region calculate a *bid* for priority. The bid is based on the positions and speeds of vehicles as reported over WiFi. At the junction a *junction agent* assigns the green light to those sections of road with the *highest bid*. Coordination between junctions is achieved through a *zone agent* which can re-weight bids to encourage coordination. In simulation tests the auctioning agent system using WiFi data outperformed the MOVA control system (Vincent and Peirce, 1988), which uses inductive loops.

The *human trained neural network* control system (Box and Waterson, 2012) uses the same system of bids as the auctioning agent system. However instead of using the bids as proxies for priority it uses the set of bids as an abstract simplified *state* describing the situation at the junction. A neural network is

used to classify the resulting state space into decisions, namely which stage of the junction gets the green light. The training data for the network is provided by a human expert when they control the simulated junction via a computer game interface. Thus the human trained neural network is a machine learning junction control system that learns strategies from a human expert. In simulation tests the human trained neural network outperformed the auctioning agent control system.

The above control systems were both developed and tested on a simulation test-bed. This uses SIAS-Paramics micro-simulation software to simulate the movement of vehicles through the network. SIAS-Paramics is connected with a number of specially developed software modules to simulate sensor data, make control decisions and implement the control (i.e. change the traffic light colour) in the simulation. The same test-bed has been used in the research presented in this paper. It is described in full in Box and Waterson (2010b,a, 2012).

There are two principal shortcomings to using human experts to train machine learning junction controllers. Firstly, to implement this in practice would be costly because human time is relatively expensive. Secondly the best possible performance of the system is limited to being as good as the human trainer. These shortcomings motivate the investigation into extending the *supervised learning* approach of the human trained neural network to build a *reinforcement* trained neural network.

As already described, junction control systems use measurements to determine the state on the road in order to make control decisions. However the state on the road right now tells us something about the decisions that were made in the past. In principal the controller can evaluate whether decisions made in the past were good or bad and learn from *these* data just as it learns from the data generated by the human trainer. This is the approach of *temporal difference* learning.

Research in other applications of artificial intelligence has shown that problems that can be solved using a neural network trained by supervised learning can also respond well to a neural network trained by temporal difference learning. A well known example of this is the work of Gerald Tesauro (Tesauro, 2002) who developed the computer Backgammon program *Neurogammon*, which employed a neural network to learn strategies from human expert backgammon players. He then went on to develop "TD-gammon", a Backgammon program that used a neural network trained by temporal difference (TD) learning in simulations where the program competed against itself.

In this paper we present an adaptation to the neural network based junction control system described in Box and Waterson (2012). This adaptation enables the controller to be trained under simulation by temporal difference reinforcement learning. The principal contributions of the paper are as follows.

1. A new machine learning junction controller, which employs a two layer neural network to learn strategies through temporal difference reinforcement learning.
2. A comparison between the performance of the human trained junction

controller and the TD trained junction controller in simulation tests.
3. Simulation tests of performance on both a simple isolated T-junction, and a pair of closely spaced junctions, where coordination is necessary.

2. Machine learning junction control strategies

2.1. Overview

2.1.1. Bids

When simulating GPS + WiFi data from vehicles we can collect estimates of the position and speed of every vehicle in the simulation. At any given time these data describe the *state* of the network. To make the problem more tractable and to speed up calculation time this raw description of the state is simplified in a pre-processing operation that generates *bids*.

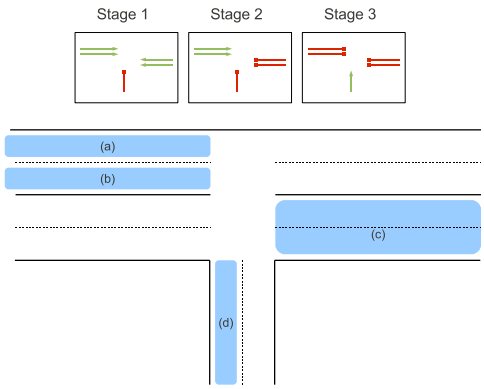

Figure 1: Schematic of the Simple-T model showing the bid zones and the junction staging

To affect this the road network around the junction is divided into regions. Figure 1 shows a schematic of one of the junctions discussed in this paper with the regions marked $(a - d)$. Each region is monitored by an agent, which calculates a *bid* based on the position and speed data of the vehicles within that region. The bid is calculated using

$$B = \sum_{c \in C} 1 - \alpha V_c - \beta X_c \tag{1}$$

Here C is the set of all vehicles monitored by the lane agent; V_c is vehicle speed and X_c is the distance of the vehicle from the junction; α and β are coefficients that can be tuned to adjust the relative influence that the number of vehicles, the vehicle speed and the vehicle distance each have on the size of the bid. In previous work (Box and Waterson, 2010b) it has been shown that (assuming S.I. units are used) values of $\alpha = 0.01\, sm^{-1}$ and $\beta = 0.001\, m^{-1}$ provide a good balance between influences. These values were adopted in this work.

The term "bid" is used because this method was first employed by the auctioning agents signal control algorithm (Waterson and Box, 2012) where this bid was designed to be indicative of the need for priority on a section of road. For example more vehicles increases the bid, slower moving vehicles increases the bid and vehicles closer to the end of the road section increases the bid (and vice-versa). In the work presented here the set of bids from each of the regions is simply a description of the state of the road network. Therefore it is unimportant that the bid is not a true representation of the need for priority. However if it is at least reasonably representative it can make the classification task easier (Box and Waterson, 2012). In the example shown in Figure 1 there are four regions and thus the state is described by four bids.

2.1.2. Training and using the neural network

Figure 2 shows a process flowchart of the system described in this paper. At each time step the simulator outputs raw measurements of the state on the road. This is then processed to generate bids, these form the input units to the neural network. The neural network then calculates which stage of the simulated junction should have the green light and passes this information to the simulator. The time step used in this paper is $\delta t = 10\,s$. This time step is quite course and could be reduced, however the results presented here and in earlier work (Box and Waterson, 2012) demonstrate good comparative performance using this time step.

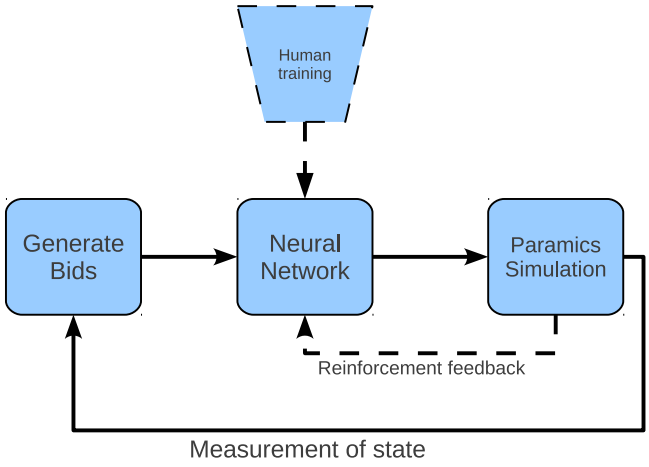

Figure 2: Flowchart showing the process for controlling the simulation using a neural network (solid arrows) and the processes for training network's parameters (dashed arrows)

The decision of which stage to give the green light is governed by the values of the weights in the neural network these weights can be trained either by

human input or by TD learning.

Under human training a human controls the simulation through a computer game interface, choosing which stage to give the green light every δt seconds. Every time the human makes a decision this generates a *pattern* linking a set of bids (*state*) to a stage decision. These patterns are then used to train the network as described in Section 2.2.

Under TD learning information contained in the raw state measurement at time step t is used to evaluate the decision made at time step $t-1$. Feedback is provided to the neural network to adjust the values of its weights accordingly. This is described in detail in Section 2.3

2.2. Supervised learning with a human expert

For each pattern recorded there is a list of bids. A unit "offset" element is appended to this list to create the J dimensional bid vector $\mathbf{b}$ (where J is the number of bids plus one). For each pattern there is also a K dimensional target vector $\mathbf{t}$ with elements $t_k \in \{0, 1\}$, where $\sum_{k \in K} t_k = 1$.

Layer 1. The first layer of the neural network forward propagation transforms the bid vector $\mathbf{b}$ onto an H dimensional *hidden units* vector $\mathbf{z}$ using the following transformation.

$$z_h = \tanh\left(a_h^{(1)}\right) \tag{2}$$

$$\mathbf{a}^{(1)} = \mathbf{W}^{(1)}\mathbf{b} \tag{3}$$

where $\mathbf{W}^{(1)}$ is a $H \times J$ matrix of weights (to be learned).

The length of vector $\mathbf{z}$, called the *number of hidden units*, is a variable that can be tuned to control the complexity of the transformation. A low number of hidden units can limit the complexity of the transformation giving a poor fit to the evidence data. A high number of hidden units allows more complexity in the transformation but can lead to over fitting of the data.

Layer 2. The second layer of the neural network forward propagation substitutes the transformed bid $\mathbf{z}$ into the softmax function (Bishop, 2006).

$$y_k = \frac{\exp(a_k^{(2)})}{\sum_{p \in K} \exp(a_p^{(2)})} \tag{4}$$

$$\mathbf{a}^{(2)} = \mathbf{W}^{(2)}\mathbf{z} \tag{5}$$

where $\mathbf{W}^{(2)}$ is a $K \times H$ matrix of weights to be learned.

The weights of the network are learned by attempting to minimizing the error between the human generated training data and the network output y_k. This is done using back propagation and gradient descent. The method is detailed in (Box and Waterson, 2012). In effect the neural network above parametrizes K functions (y_k) (recall K is the number of stages), each of these functions can be described as something like: the probability that the human expert would pick stage k given the current bids $\mathbf{b}$.

2.3. Reinforcement learning by temporal difference

Reinforcement learning involves learning the combinations of states and decisions that maximize some cumulative numerical reward. This approach differs from the supervised learning approach described above because there is no "trainer" providing explicit examples of correct state-decision combinations, rather learning happens on-line via an exploration of the state-decision space and the receipt of feedback (Sutton and Barto, 1998). Therefore reinforcement learning can be said to be learning through experience.

Temporal Difference (TD) is a method for reinforcement learning (Sutton and Barto, 1998; Sutton, 1988). In large continuous state spaces this approach employs a *value function* (discussed below) representing the nominal "value" of decisions at points in state space. At each step in the learning process the value of the previous state is adjusted according to both the feedback received *and* the value of the current state. Thus adjustments to the value function are made even if the receipt of feedback is delayed (bootstrapping).

The TD learning employed here uses the same two layer neural network described in Section 2.2. Under TD learning the human is no longer involved so we could describe the functions y_k as simply the probability that k is the right stage to pick given the current bids $\mathbf{b}$. However, although the values of the K functions will sum to unity across the bid space, we have no reason to believe that the learning process is driving the functions towards valid probabilities so it is usual in reinforcement learning literature to refer to the function y_k as the "value" of picking stage k for any given combination of bids $\mathbf{b}$. Thus the functions y_k become the *value functions* for TD learning (discussed above) and the neural network parametrizes these functions.

2.3.1. Q-learning

Under temporal difference learning the value of a stage decision at a particular state, $y_k(\mathbf{b})$ is adjusted upwards or downwards slightly according to feedback received after that state-decision combination is encountered in training. The algorithm used for doing this in this paper is the Q-learning algorithm (Watkins, 1989), shown below.

$$y_k(\mathbf{b}_t) \leftarrow y_k(\mathbf{b}_t)(1-\alpha) + \alpha\left(R_{t+1} + \gamma \max_k y_k(\mathbf{b}_{t+1})\right) \tag{6}$$

where R is the feedback signal. Parameter α is the learning rate, which can be tuned to determine how heavily the functions are adjusted after each decision. Parameter γ is the discount factor, which adjusts how much relative weight is given to the feedback from the last decision compared to the long term movement of the function due to all the feedback received. According to (6) at each step where a decision is made the value of the state-decision function from the previous step $y_k(\mathbf{b}_t)$ is adjusted as a function of the feedback received since the last decision and the maximum value of the current state-decision. We specify here $\max_k y_k(\mathbf{b}_{t+1})$ because under reinforcement learning we do not automatically select the stage with the highest value as we do with supervised learning. This is discussed further in Section 2.3.3.

2.3.2. Back-propagation

The Q-learning algorithm shown in (6) assumes that the values ($y_k(\mathbf{b}_t)$) can be adjusted directly, however this is not the case when using a neural network to parametrize the functions. Here we must affect the change in value by adjusting the weights ($\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$) of the neural network. This is done through back-propagation.

$y_{c,t}$ is defined as the element of $\mathbf{y}_t$ which corresponds to the chosen stage ($k = c$). The gradient of $y_{c,t}$ with respect to the two weights matrices is given by

$$\nabla y_{c,t}(\mathbf{W}^{(1)}) = \boldsymbol{\delta}_t^{(1)} \mathbf{b}_t^{\mathsf{T}} \tag{7}$$

$$\nabla y_{c,t}(\mathbf{W}^{(2)}) = \boldsymbol{\delta}_t^{(2)} \mathbf{z}_t^{\mathsf{T}} \tag{8}$$

where $\boldsymbol{\delta}_t^{(2)}$ is given by

$$\boldsymbol{\delta}_t^{(2)} = (\mathbf{I}_t - \mathbf{Y}_t)\, y_{c,t} \tag{9}$$

$\mathbf{I}$ is a K dimensional vector with elements $I_k = 1$ if $k = c$ and $I_k = 0$ otherwise. $\boldsymbol{\delta}_t^{(1)}$ has H elements

$$\delta_{t,h}^{(1)} = \left(\mathbf{W}^{(2)} \boldsymbol{\delta}_t^{(2)}\right)_h \left(1 - z_{t,h}^2\right) \tag{10}$$

To update the weights of the neural network the following algorithm is used.

$$\mathbf{W}^{(1)} \leftarrow \mathbf{W}^{(1)} + \mathbf{E}^{(1)} \alpha (R_{t+1} + \gamma \max_k y_{k,t+1} - y_{c,t}) \tag{11}$$

where $\mathbf{E}^{(1)}$ is given by.

$$\mathbf{E}^{(1)} \leftarrow \mathbf{E}^{(1)} \lambda \gamma + \nabla y_{c,t}(\mathbf{W}^{(1)}) \tag{12}$$

The purpose of the first term in (12) is to convey reward back to previous decisions, other than the immediately previous one, in amounts that decrease as decisions stretch further back in time. The λ coefficient controls how quickly the reward decays. In the limiting case when $\lambda = 0$ (12) becomes $\mathbf{E}^{(1)} = \nabla y_{c,t}(\mathbf{W}^{(1)})$ and reward is assigned to the previous decision only. The purpose of setting λ above zero is to recognize that multiple decisions leading up to the receipt of reward may deserve "credit". The same process in (11) and (12) is performed to update $\mathbf{W}^{(2)}$.

2.3.3. Stage selection strategy

The above back-propagation of feedback will adjust the value attached to combinations of bids and stage decisions. In the supervised learning case, when the network has been trained, then in operation the stage with the highest probability (or value) is always selected. In reinforcement learning literature this is called a "greedy" strategy. This is not so helpful when learning by temporal difference as it can leave areas of bid-decision space unexplored. To encourage the junction controller to try out new options while learning by TD a strategy called $\epsilon -$ greedy is employed. Here the greedy strategy is employed most of the time but occasionally (with some probability ϵ) the stage decision is made at random.

2.3.4. Feedback signal

In temporal difference learning the feedback signal (R_{t+1} term in equation (6)) is designed to give positive "reward" if the junction controller makes a good decision and negative reward if a bad decision is made. The reward signal used in this paper is calculated as a function of total vehicle lifetime L. A vehicle is defined to be "born" when it enters a fixed region surrounding the junction (in this paper this is simply the limits of the simulated area). The vehicle "dies" when it crosses the junction stop line. Therefore the total lifetime at time t (L_t) is the sum of the ages in seconds of all vehicles "alive" at time t.

The reward term R_{t+1} is a measure of the change in lifetime between one time-step and the next. It is calculated as

$$R_{t+1} = 2 - \frac{2L_{t+1}}{L_t} \tag{13}$$

and is bounded using

$$-2 \leq R_{t+1} \leq 2 \tag{14}$$

Therefore events which lead to a reduction in lifetime such as the release of a queue of vehicles at a junction will receive a positive feedback. Events that lead to an increase in lifetime such as building up a queue will receive negative feedback.

3. Simulation experiments

3.1. Junction models

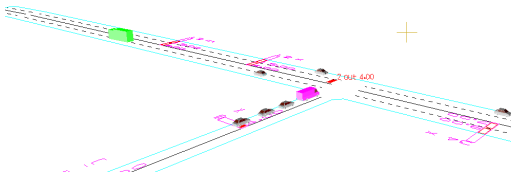

Figure 3: Simulation screen shot of the simple T-Junction model

The simulation experiments presented in this paper used two junction models. The first (shown in Figure 3) is a simple t-junction (Simple-T). The second (shown in Figure 4) is a model of the high road area in Southampton, UK, which contains two signalised junctions a short distance apart (High-Rd).

Figures 1 and 5 show schematics of the Simple-T model and the High-Rd model respectively. The schematics indicate the road regions that are covered by the lane agents monitoring the junctions. They also indicate which lanes are given the green light in each of the junction stages.

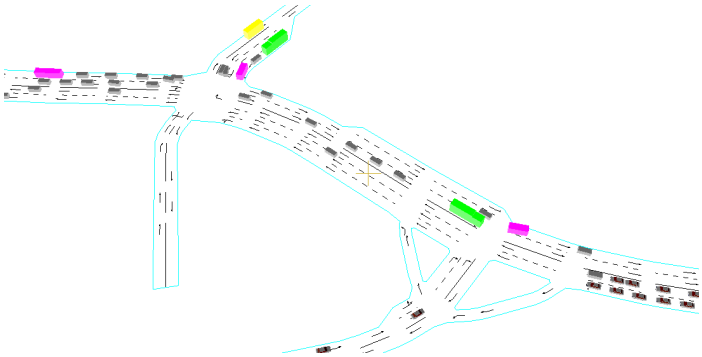

Figure 4: Simulation screen shot of the High-Rd model

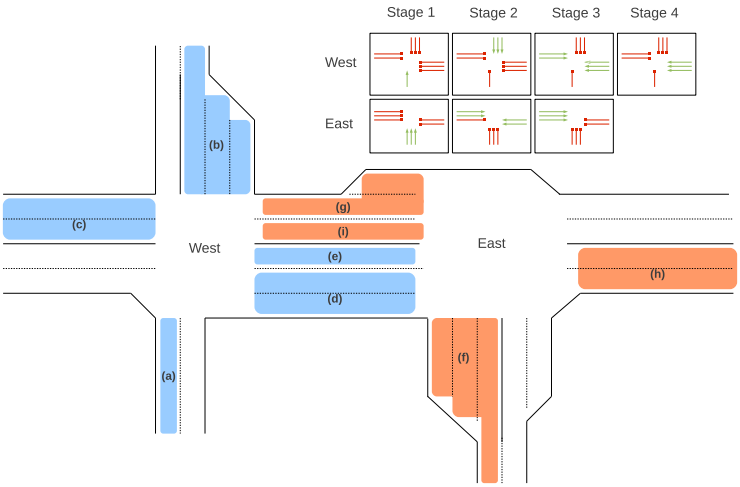

Figure 5: schematic of the High-Rd junction showing the bid zones and the junction staging

3.1.1. Demand

The demand in the simulation is set as the number of vehicles per minute that flow between origin and destination zones. There is one zone at the end of each road in the model. At any time the demand between zone i and and zone j is calculated as

$$D_{i,j} = d_{i,j} d_t \tag{15}$$

where $d_{i,j}$ is a base level of demand between zones i and j and d_t is a demand multiplier. This coefficient can vary over the course of a simulation. The base level demand $d_{i,j}$ between each of the origin and destination zones for the Simple-T model and the High-Rd model are shown in Tables 1 and 2 respectively. In the tables the zones are labelled using appropriate compass points.

	W	E	S
W	–	18.96	5.0
E	24.01	–	1.26
S	4.05	4.05	–

Table 1: Basic demand matrix (vehicles per minute) for the Simple-T model

	W	SE	E	N	SW
W	–	4.81	8.76	1.95	0.19
SE	1.95	–	3.89	1.95	0.049
E	8.76	6.89	–	0.97	0.097
N	2.92	0.97	2.92	–	0.097
SW	0.097	0.097	0.097	0.097	–

Table 2: Basic demand matrix (vehicles per minute) for the High-Rd model

3.1.2. Neural network structure

Each of the modelled signalized junctions, one in the Simple-T model and two in the High-Rd model are controlled by a neural network. The standard structure is a two layer network with J input units H hidden units and K output units. Recall from Section 2 that J is the number of bids that describe the state of the network plus one and K is the number of signal stages. The number of hidden units H is a variable that can be tuned to control the complexity of the approximating function. The values of H employed in this paper are the same as those used for neural networks designed for training by a human (Box and Waterson, 2012).

In the Simple-T model's network $J = 5$, $K = 3$ and $H = 7$. In the network controlling the East junction in the High-Rd model $J = 10$, $K = 3$ and $H = 11$. For the West junction $J = 10$, $K = 4$ and $H = 11$. Note that the networks controlling the East and West Junctions in the High-Rd model take data from all the lane agents in the model as inputs. This allows each junction to "know" the state of the other and, although operating independently, coordination can be an emergent property (Box and Waterson, 2012).

3.2. Training phase

The training phase for human training is described in detail in (Box and Waterson, 2012) here we discuss only the training under TD learning as laid

out in Section 2.3.

From Section 2.3 there are four parameters which must be set ahead of TD training. These are the discount factor γ, the reward decay rate λ, the learning rate α and the probability of random stage selection ϵ. Through a testing process where these parameters were systematically varied individually values of $\gamma = 1$ and $\lambda = 0.1$ were selected on the basis of best performance. Parameters α and ϵ were varied in the three stages of training described below.

The training phase was divided into three stages. To begin with, the weights of the neural networks were initialized randomly. Training took place over repeated simulations of fixed duration S with a constant level of demand defined by the demand multiplier d_t. Values of S and d_t were varied during the training stages as described below.

Training Stage 1. Values $\alpha = 0.1$, $\epsilon = 0.1$ and $S = 15$ mins were used. The demand multiplier was initially set to $d_t = 0.2$ and was thereafter increased evenly in increments of 0.2 up to $d_t = 1.0$. The policy of repeatedly restarting the simulation every 15 mins and gradually increasing the demand is useful in the early stages of training because with very naive strategies queues in the simulation can build up rapidly. Therefore a lot of time can be spent learning how to discharge large queues before the it is possible to learn strategies under more normal conditions.

Training Stage 2. Values $\alpha = 0.1$, $\epsilon = 0.1$ and $S = 30$ mins were used. Now d_t was cycled more rapidly varying as before but incrementing at every simulation restart.

Training Stage 3. Was as training stage 2 but with $\alpha = 0.01$ and $\epsilon = 0.01$. The reduction in the learning rate and the probability of random stage selection is designed to reduce the exploration of different strategies and encourage more stable learning and improvement on the current strategy.

The duration of the three training stages was different for training on the Simple-T model and the High-Rd model. Table 3 shows the durations used.

	Time (simulated hrs)	
Stage	Simple-T	High-Rd
Stage 1	90	580
Stage 2	75	360
Stage 3	85	560

Table 3: The times in simulated hours of the three stages of training used on each of the test models.

3.3. Testing phase

Once the parameters of the neural networks had been trained they were tested on the Simple-T and High-Rd models with the training mode disabled.

Test runs were up to four hours in length and during the tests the demand multiplier d_t (Equation (15)) was varied according to the plot shown in Figure 6.

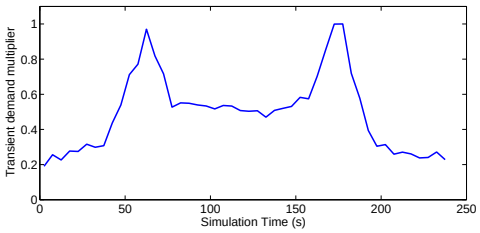

Figure 6: Variation of the transient demand multiplier throughout four hour simulation period

During the simulations S-Paramics records detailed information about the journeys of every simulated vehicle. In the analysis presented here the main measurements used are journey time t and delay θ. For a given vehicle p the time it takes to travel from its origin i to its destination j is its journey time t_p. The vehicle's free flow travel time $t_p^{(\mathrm{ff})}$ is the theoretical time that it would take to travel between i and j if it were unimpeded by other vehicles or red signals. The delay for vehicle p is the difference between these two times.

$$\theta_p = t_p - t_p^{(\mathrm{ff})} \tag{16}$$

4. Test results

There is no consensus in the literature for what constitutes optimal performance for a signalized junction controller. A number of metrics can be used to judge performance including vehicle flow rates, speeds, journey times or even emission levels. In this paper we broadly assume optimal performance to mean minimizing delay (θ) averaged across journeys and time periods and also minimizing the variance of the distribution over journey times. The latter is important because it is an indicator or equitability. A distribution with low variance indicates that all vehicles are receiving reasonably equal treatment. A distribution with higher variance indicates that some vehicles are getting through the junction more quickly than others.

4.1. Performance

After 260 simulated hours of training the neural network controlling the Simple-T model was tested in a four hour test as described above. The mean delay measured over all journeys in this test was 14.36 seconds. After 1500 simulated hours of training the neural network controlling the High-Rd model was tested in the same way. The measured mean delay was 20.82 seconds. Table 4 compares these values of mean delay with those measured in otherwise

	Delay (s)	
Model	Human trained	TD trained
Simple-T	14.12	14.36
High-Rd	17.44	20.82

Table 4: Delay averaged over all journeys during 4-hour simulation tests on the two test junctions, using various control strategies.

identical tests where the controlling neural networks were trained by a human expert as described in Box and Waterson (2012). These data show that the performance of the temporal difference trained neural network is very similar on the Simple-T model, but the delay is 3 seconds higher on the High-Rd model. The large amount of training and the lower performance of the High-Rd neural network are indicative of the increased complexity of this model where two junctions need to be coordinated.

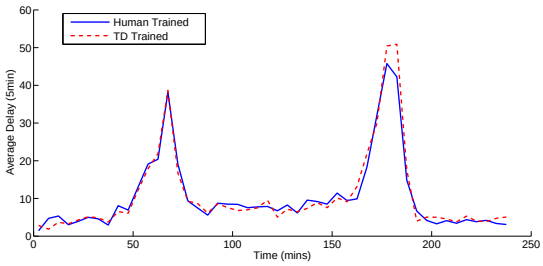

Figure 7: Variating in delay duting the test on the Simple-T model

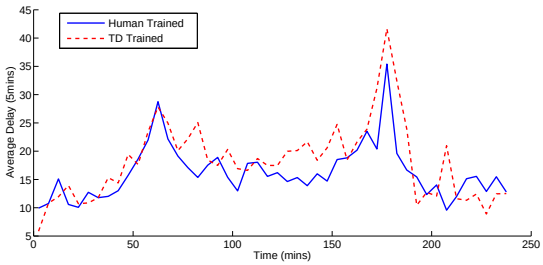

Figure 8: Variating in delay duting the test on the High-Rd model

Figures 7 and 8 show how the delay varied over the course of the tests for the Simple-T and High-Rd models respectively. Here the values plotted at each time are the measured delay over the previous 5 minutes. Figure 7 shows that the performance of the human trained and temporal difference (TD) trained networks are very close over the course of the test with the exception of the

second peak in delay. Figure 8 shows a similar trend but with the human trained network more consistently outperforming the TD trained network in the middle 2 hours of the test.

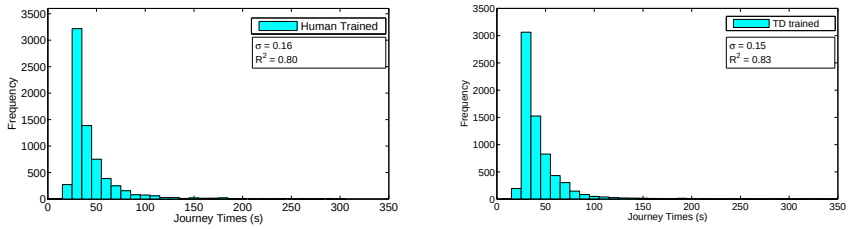

Figure 9: Journey time histograms for the tests on the Simple-T junction model where the neural network was trained by a human (*left*) and by temporal difference (*right*)

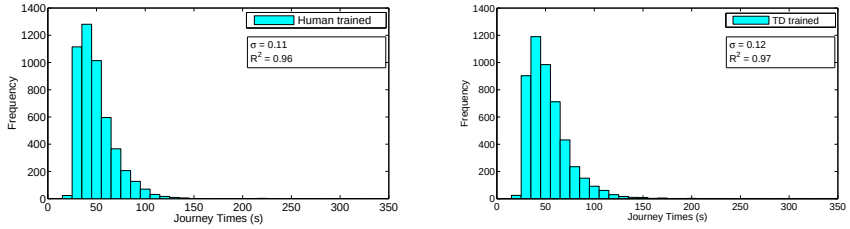

Figure 10: Journey time histograms for the tests on the High Rd junction model where the neural network was trained by a human (*left*) and by temporal difference (*right*)

While delay is a useful indicator of a junction controller's performance it is also important to consider equitability, indicated by the variance of the journey time distributions. Figure 9 shows the journey time distributions for the tests on the Simple-T model. This shows that while human training very slightly outperformed TD training in terms of delay, TD training very slightly outperforms human training in terms of equitability with a variance of 0.15 vs 0.16 in the journey time distributions. Figure 10 shows the journey time distributions for the High-Rd model. Here the human training is very slightly more equitable than the TD training but the difference is again marginal.

4.2. Analysis of training

To investigate how the performance of the TD trained neural network evolved over the course of the training we can take samples of the (partially) trained network weights that were recorded throughout training. These semi-trained weights were tested on short (20 minute) tests using the demand profile in Figure 6 between 165 and 185 minutes. Figure 11 shows the total mean delay for 350 tests with weights sampled over the first 80 hours of training on the Simple-T model. This shows that, as expected, performance is very bad for the initial randomly initiated weights. Delay in the first test is ~ 210 seconds. This

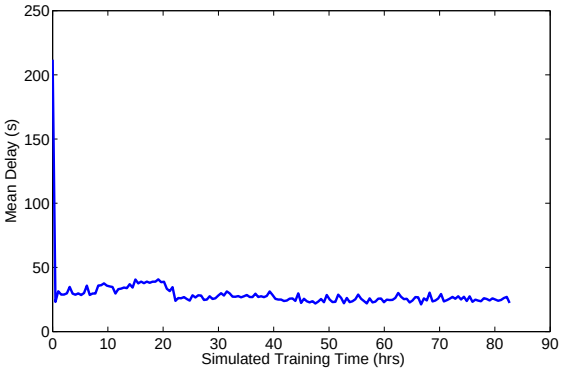

Figure 11: The improvement in performance of the TD trained network over part of the training period

very quickly improves with delay dropping down below 50 seconds in the first hour. Thereafter performance is characterised by long periods of stagnation punctuated by sudden "breakthroughs" such as the steps in delay reduction at around 20 hours and 40 hours.

4.3. Journey time distribution shape

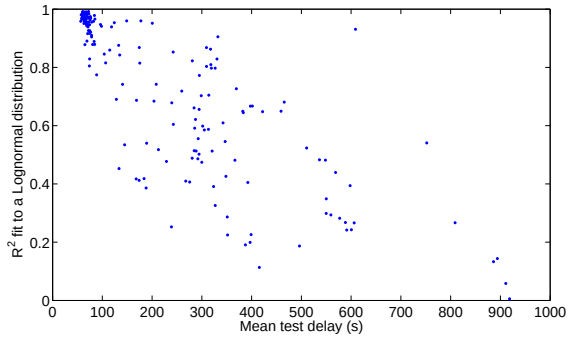

Figure 12: Scatter points for tests on the High-Rd model using parameter weights from different stages of the training process. The mean delay recorded in each test is plotted against the R^2 value of the fit of the journey time distribution to an ideal logmormal distribution.

Despite similarities in the approach between TD learning and Dynamic Programming it is not possible (for the system presented here) to analytically prove convergence on an optimal strategy (Sutton and Barto, 1998).

We can expect the optimal control strategy to produce a journey time distribution with mean as close as possible to the free flow travel time, and variance as low as possible. It follows that the journey time distribution under optimal control will have a defined mean, variance and *shape*.

Previous research has demonstrated that the journey time distribution for vehicles on a *link* (that is a stretch of road between junctions) can be well approximated by the lognormal distribution (Montgomery and May, 1987; Rakha et al., 2006) and this fact has been employed in many models of traffic flow (Kaparias et al., 2008). The evidence for the journey time distribution across junctions is less well established, however it is reasonable to assume that under perfect control the journey times experienced by vehicles should differ by as little as possible from the condition of free flow on a link.

In fact the evidence from the results in Section 4.1 shows that the journey time distributions across the junctions *are* a good fit to the lognormal distribution. The R^2 value of the fit to an ideal lognormal distribution is shown on each plot in Figures 9 and 10.

To examine whether there is a relationship between the performance of a control strategy and its fit to the lognormal distribution we measured the R^2 value of the journey time distributions for the same 20 minute tests described in Section 4.2, but that were carried out on the High-Rd model. Figure 12 shows the measured R^2 value in these tests plotted against the performance in terms of total mean delay. These data indicate a trend that the better performing strategies are also a closer fit to the lognormal distribution.

5. Conclusions

The results presented in Section 4 show that temporal difference reinforcement learning can be used to train a junction controller from a position of no prior information, where it is changing the lights at random, to a position of high performance.

On the Simple-T junction model the temporal difference trained neural network matched the performance of the human trained neural network in terms of both delay and equitability. In previous work (Box and Waterson, 2012) the human trained neural network has been shown to significantly outperform the auctioning agent junction controller (Waterson and Box, 2012) and the MOVA junction controller (Vincent and Peirce, 1988).

On the more complex High-Rd model TD training matched human training in terms of equitability and exhibited lower performance in terms of delay with on average 3 seconds more delay per vehicle. The High-Rd model was also subject to ~ 6 times more training in terms of simulated hours. This is indicative of the increased complexity of the problem to be solved in the High Rd model where two closely spaced junctions must be coordinated for best performance.

The performance improvement of the neural networks during TD training was characterized by long periods of stagnation punctuated by "breakthroughs" where performance suddenly increased (Figure 11). This makes it difficult to know if the best possible performance under TD training has been reached and performance may improve further with more training.

The motivations behind investigating TD learning (Section 1.2) were to find an alternative to human training and to potentially outperform human training.

While the former has been achieved the latter has not. Future investigations to improve the performance of TD learning will include more training and use of different feedback signals (Section 2.3.4). However it is of course possible that the performance of the human trained network in this system is very close to optimal.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer.

Box, S., Waterson, B., 2012. An automated signalized junction controller that learns strategies from a human expert. Engineering Applications of Artificial Intelligence 25(1), 107–118.

Box, S., Waterson, B.J., 2010a. Comparison of signalized junction control strategies that use localization probe data, in: The IET Road Transport Information and Control Conference, 25 - 27 May 2010, Dexter House, London, UK.

Box, S., Waterson, B.J., 2010b. Signal control using vehicle localization probe data, in: 42nd Annual UTSG Conference, University of Plymouth, 5-7 January 2010.

Chen, C., Heydecker, B., 2009. Adaptive traffic signal control using approximate dynamic programming, in: UTSG conference, London, January, 2009.

Choy, M.C., Srinivasan, D., Cheu, R., 2003. Cooperative, hybrid agent architecture for real-time traffic signal control. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 33, 597–607.

COOPERS, 2010. Co-operative systems for intelligent road safety. Online at http://www.coopers-ip.eu/.

Heung, T.H., Ho, T.K., Fung, Y.F., 2005. Coordinated road-junction traffic control by dynamic programming. Intelligent Transportation Systems, IEEE Transactions on 6, 341–350.

Heydecker, B., Cai, C., Wong, C., 2007. Adaptive dynamic control for road traffic signals, in: Networking, Sensing and Control, 2007 IEEE International Conference on, pp. 193–198.

Hunt, P., Bretherton, R., Robertson, D., Royal, M., 1982. Scoot on-line traffic signal optimisation technique. Traffic Engineering and Control 23, 190–192.

Kaparias, I., Bell, M.G., Belzner, H., 2008. A new measure of travel time reliability for in-vehicle navigation systems. Journal of Intelligent Transportation Systems 12, 202–211. `http://www.tandfonline.com/doi/pdf/10.1080/15472450802448237`.

Kompfner, P., 2008. Cvis – cooperative for mobility. Online at http://www.cvisproject.org/download/cvis_brochure_May2008_Final.pdf.

Mikami, S., Kakazu, Y., 1993. Self-organized control of traffic signals through genetic reinforcement learning, in: Intelligent Vehicles '93 Symposium, pp. 113–118.

Montgomery, F., May, A.D., 1987. Factors affecting travel times on urban radial routes. Traffic Engineering and Control 28, 452–458.

Rakha, H., El-Shawarby, I., Arafeh, M., Dion, F., 2006. Estimating path travel-time reliability, in: Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE, pp. 236 –241.

SAFESPOT, 2010. Cooperative vehicles and road infrastructure for road safety. Online at http://www.safespot-eu.org/.

Sreedevi, I., 2005. ITS decision – services and technologies – loop detectors. Available online at: http://www.calccit.org/itsdecision/serv_and_tech/Traffic_Surveillance/road-based/in-road/loop_summary.html .

Sutton, R.S., 1988. Learning to predict by the methods of temporal differences. Machine Learning 3, 9–44. 10.1007/BF00115009.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning An Introduction. Adaptive Computation and Machine Learning, MIT Press.

Tesauro, G., 2002. Programming backgammon using self-teaching neural nets. Artificial Intelligence 134, 181–199.

Vincent, G., Peirce, J., 1988. 'MOVA': Traffic responsive, self-optimising signal control for isolated intersections. TRRL Research Report RR170.

Waterson, B.J., Box, S., 2012. Quantifying the impact of probe vehicle localisation data errors on signalised junction control. Accepted for publication in IET Intelligent Transportation Systems .

Watkins, C., 1989. Learning from delayed rewards. Ph.D. thesis. University of Cambridge.

Wood, K., Crabtree, M., Gutteridge, S., 2006. Pedestrian and vehicular detectors for traffic management and control. TRL Report .