Vehicle Traffic Light Control Using SARSA

Thomas L. Thorpe

April 2, 1997

Abstract

SARSA (Sutton, 1996) is applied to a simulated, traffic-light control problem (Thorpe, 1997) and its performance is compared with several, fixed control strategies. The performance of SARSA with four different representations of the current state of traffic is analyzed using two reinforcement schemes. Training on one intersection is compared to, and is as effective as training on all intersections in the environment. SARSA is shown to be better than fixed-duration light timing and four-way stops for minimizing total traffic travel time, individual vehicle travel times, and vehicle wait times. Comparisons of performance using a constant reinforcement function versus a variable reinforcement function dependent on the number of vehicles at an intersection showed that the variable reinforcement resulted in slightly improved performance for some cases.

1. Introduction

A variety of traffic control strategies are being studied in real traffic networks and in simulation. The Denver Regional Council of Governments works with the Colorado Department of Transportation and citizens to identify and modify problem intersections (Garnaas, 1996). Computers are used to monitor the traffic flows for critical intersections throughout the Denver region. The computers have the capability to change traffic light timing remotely but are only used to collect data for traffic analysis. Recently a major traffic artery was re-timed from 90 seconds in the heavy traffic flow direction to 100 seconds. This resulted in an 87% reduction in times stopped at lights. Stockholm, Sweden, uses remote television cameras to monitor high traffic flow areas (Olsson, 1996). The traffic conditions are directly observed and speed limits and traffic light timing can be slowly adjusted remotely. Vehicles can also be rerouted remotely to reduce congestion. Neural networks that use traffic density approximations to

indicate ideal vehicle speed within sections of a lane of traffic have been simulated, achieving good results (Ho and Ioannou, 1996). This method does not try to control traffic lights and requires sampling along a section of highway where there are no on-ramps or off-ramps.

Current traffic controllers in wide use are very primitive and require frequent manual adjustments to keep traffic flowing smoothly. In this article, it is shown that reinforcement learning with complete knowledge of vehicle locations approaches the best traffic light performance that can possibly be achieved for the limited traffic simulation used.

Section 2 provides an overview of reinforcement learning and in Section 3, the traffic simulation and several conventional traffic light controllers are described. The application of SARSA to the traffic light control problem is described in Section 4. The results are summarized in Section 5, and Section 6 presents conclusions.

2. Reinforcement Learning

Reinforcement learning is a form of learning typically used for controlling processes. It has been applied to balancing an inverted pole (Anderson, 1988), optimizing elevator performance (Crites and Barto, 1996), determining the actions required to rock an under-powered vehicle out of a valley (Sutton, 1996), playing backgammon (Tesauro, 1995) and many others. The TD-Gammon backgammon program has been so successful that experts are learning new strategies from it.

Reinforcement learning techniques differ from supervised learning, such as error back-propagation in neural networks (Rumelhart, 1986), because neural networks require a teacher to provide answers or desired output values for a set of inputs. The errors or differences between the output of the learning agent and desired values are used to modify the network weights. After sufficient training the neural network will be able to predict the output for a set of inputs with varying accuracy. For control problems the correct actions for a given situation or state may not be known and neural networks by themselves might not solve the problem since there may not be any examples to learn from.

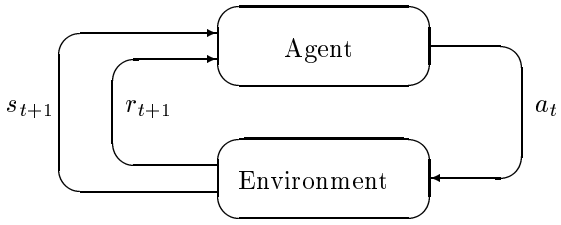

Figure 1: Reinforcement learning agent relationship to the process being controlled.

Reinforcement learning can discover the optimum actions by interacting with its environment as shown in Figure 1. By taking an action, a_t, at time, t, the agent interacts with the world producing a new state, s_{t+1}, and receiving a reward, r_{t+1}, as a result. The agent uses the reward to modify the policy it will use for choosing future actions. Learning in this way allows the agent to adapt to the environment it is controlling. If the dynamics of the system being controlled changes over time, for example increasing friction in a robots arm, the reinforcement learning agent will be able to adjust to the new environment and continue to function properly. This direct interaction with its environment allows the learning agent

to develop an optimal control strategy without requiring a model of the process it is controlling.

Reinforcement learning procedures have been shown to converge for absorbing Markov processes, meaning that the learning agent can see the state of the environment, the next state is only dependent on the current state, and that all sequences eventually terminate (Sutton, 1988).

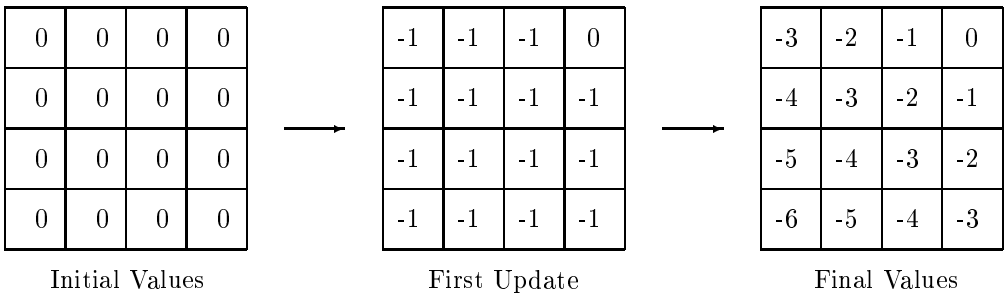

Figure 2: Evolution of state values during reinforcement learning for a simple problem.

The mechanics of reinforcement learning can be illustrated with a simple grid problem as shown in Figure 2. The grid represents the physical layout of a room split into 16 squares. The goal is to reach the upper right corner of the grid as quickly as possible. By taking one step in any vertical or horizontal direction, the occupant moves with certainty to the next grid in the direction they are moving. To begin the learning process the value of each grid or state is set to zero. After learning has proceeded substantially, the state values can be used to choose which direction to proceed, by selecting the next state with the highest possible value.

The state values can be updated using *iterative policy evaluation* which uses the following update equation:

$$V'(s_t) = r_{t+1} + V(s_{t+1}) \tag{1}$$

This says that the value of the current state is the value of the next state plus the reward received during the state transition. The value of the goal state is not updated. A common reward scheme to minimize the time required to reach the goal is: $r_{t+1} = -1$. By applying this equation to all states from left to right for the first three columns of the grid and then bottom to top for the last column, the results of the first iterative policy evaluation update are shown in Figure 2. After several iterations the final

state values are found and represent the negative number of steps to the goal when choosing the optimal action. If the update occurred in a different order, the values might be more negative and would not be optimal. This is corrected by defining the optimal state value, $V^*(s)$ as:

$$V^*(s_t) = \max_{a_t} E\{r_{t+1} + V^*(s_{t+1}) | s_t \in S, a_t \in A(s)\}. \tag{2}$$

This says the optimal value for a valid state is the maximum of the next expected state values plus reward from the available actions.

For more complex problems, the resultant state of an action might not be known ahead of time and it is more practical to base decisions on the results of actions. This is done by storing the values of action results within each state. When an action is chosen from a given state, the value for the action is updated upon receiving reinforcement from the environment. Choosing the action to implement is then a simple look up in the current state space. The value of a state action pair is denoted: $Q(s, a)$. The optimal state value is then defined as:

$$V^*(s) = \max_a Q^*(s, a), \tag{3}$$

and the optimal state-action value, $Q^*(s, a)$ is:

$$Q^*(s_t, a_t) = E_{a_t}\{r_{t+1} + \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t \in S, a_t \in A(s)\}. \tag{4}$$

The method used for the traffic controller is SARSA with eligibility traces (Sing and Sutton, 1996). SARSA is similar to equation (4), except that the "$\max Q^*$" term is replaced by the Q value for the actual action, a_{t+1}, taken at step, $t + 1$. SARSA is an abbreviation for State-Action, Reward-State-Action as implied in equation (4). Temporarily ignoring the eligibility traces the update rule is:

$$Q'(s_t, a_t) = r_{t+1} + Q(s_{t+1}, a_{t+1}) \tag{5}$$

or

$$Q'(s_t, a_t) = Q(s_t, a_t) + \Delta Q(s_t, a_t). \tag{6}$$

The delta represents the difference in values between the current state-action and the sum of the next state-action and reward. This is referred to as the *Temporal Difference error* (*TDerr*) between the

state-action pairs. The *TDerr* is given as:

$$TDerr = \alpha\{r_{t+1} + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\}, \tag{7}$$

where α is the learning rate which varies from 0 to 1. Reducing the learning rate allows the values to develop more smoothly over time during the learning process. The update rule then becomes:

$$Q'(s_t, a_t) = Q(s_t, a_t) + TDerr. \tag{8}$$

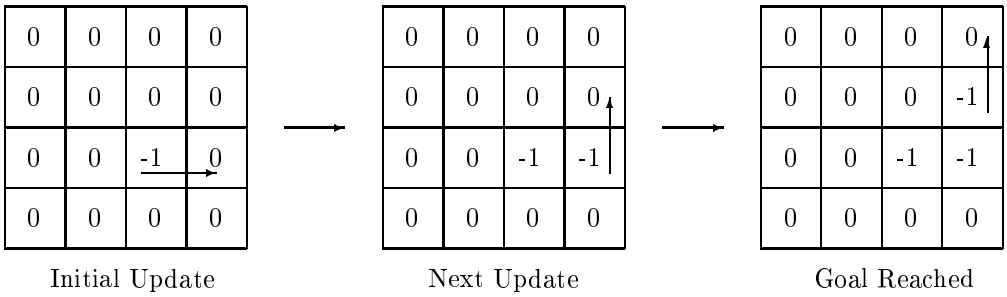

Figure 3: Update Progression without Eligibility Traces

With $\alpha = 1$, the update process from a given state to the goal is shown in Figure 3. Note: that only the action values of interest are shown in the Figure. Initially all action values are zero. The initial state is chosen randomly (the base of the arrow). The best action is chosen using an epsilon-soft policy to allow for exploration, meaning that a fraction (based on the value of epsilon, where $0 \leq$ epsilon ≤ 1) of the actions are chosen randomly amongst all the available actions regardless of their value. The action is taken and the reward and new state are observed (the point of the arrow). The next action is then chosen so the *TDerr* can be computed. In the initial update the *TDerr* is -1. The update is then applied giving the initial update results in Figure 3. The chosen action is implemented during the next time step and the process is repeated until the goal is reached concluding the current trial. A new starting state is chosen randomly and updating occurs for the next trial. Trials are repeated to sufficiently train the agent.

Eligibility traces are used to speed up the learning process by tracking the states that have been visited during a trial and adding a portion of the *TDerr* to each state-action pair visited. This also

aids the credit assignment problem, determining how much credit or blame to assign to state-action pairs for the resulting success or failure of the trial. An eligibility trace value is associated with each state-action pair. The eligibility traces are cleared to zero at the beginning of each trial and multiplied by an eligibility decay rate λ between each step in a trial. Values for λ range from 0 to 1. The eligibility trace for the state being updated is assigned a value of 1. All state-action pairs are then incremented by the product of the *TDerr* and the eligibility trace value associated with the state-action pair. This method is referred to as $TD(\lambda)$. When λ is zero, all eligibility is assigned to the previous state and zero to all other states.

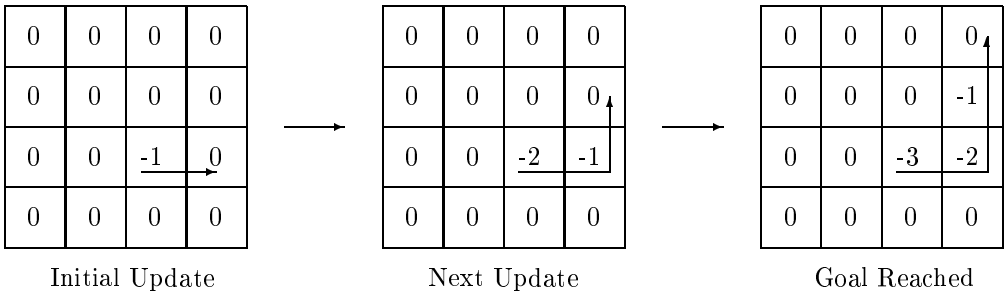

Figure 4: Update Progression using Eligibility Traces

The advantages of SARSA using $TD(\lambda)$ eligibility traces are shown in Figure 4. Both α and λ are set to 1 in this example. The *TDerr* is again -1 and the initial update gives the same result without eligibility traces. Subsequent steps not only update the current state-action pair but also the state-actions that have already been visited during the trial. The *TDerr* calculated for the current state-action pair transition is used for all of the updates along the trace for the current step. This helps speed up the learning process considerably.

3. Traffic Simulator and Control Strategies

The traffic simulator, as described by (Thorpe, 1997), uses one-second, discrete time steps for the traffic-light controller and the simulation of vehicle movement. The simulator is very realistic: the physics of motion of the traveling vehicles are closely approximated, vehicles maintain safe following distances, vehicles are stopped at red lights, and vehicles yield right-of-way, as appropriate, before entering intersections. From a green light the controller must cycle through a yellow phase for two seconds and an all-red phase for 1 second before switching the right-of-way. The only exception to the last rule is for the all-green strategy described below. The simulation takes place in a 4 x 4 grid of east-west and north-south two-lane streets (one lane in each travel direction). The simulated distance between streets is 440 feet apart. The speed limits on each lane vary from 20 to 40 miles per hour and vehicles do not exceed these limits.

Cars are inserted in the network with their starting and destination intersections chosen randomly before the simulation is started. The routes are chosen using an Iterative Deepening A* algorithm. The route for each vehicle may require more than one turn because the route selection heuristic is the travel time through a lane, which is calculated by dividing the lane length by the lane speed limit. This heuristic will drastically under-estimate the travel times when the traffic simulator is congested using high volume traffic loads. By using the same random seed for all testing, vehicles follow the same routes. This ensures that the testing results between the reference and experimental implementations are directly comparable.

Four fixed, control strategies were used to compare the SARSA results against. The *all-green* strategy sets all lights in all directions to green. Vehicles do not stop at intersections to avoid other traffic; they are allowed to travel through each other in the virtual simulator. Therefore, the all-green strategy is not physically realizable, but it does provide a baseline of best-possible performance to which other strategies can be compared. The *four-way stop* strategy is based on standard four-way stop signs: vehicles must come to a complete stop and right-of-way is passed to vehicles in a circular manner. The *fixed-duration* strategy cycles through green, yellow and red lights at predefined intervals. This is the normal type of traffic control used today. The duration can be varied but is usually based on time of day and must be

preset manually. The *greatest-volume* strategy sets to green the traffic lights in the east-west or north-south lanes depending on which direction contains the most vehicles. The lights in the other direction are set to red (after cycling through the required light phases).

For each strategy, tests were performed using 100, 500 and 1,000 vehicles. The environment is lightly loaded with 100 vehicles, starts to congest around 500 vehicles and is very congested with 1,000 vehicles. The metrics for 100, 500 and 1000 vehicles were summed to provide a combination metric. The combination metric is then searched to find the best performance when considering all three traffic loads. The simulation starts without any vehicles in the environment. The routes for all vehicles are calculated and the vehicles are queued at the block where they will enter the simulation. When the simulation begins, vehicles enter the block 100 feet from the beginning of the block when there are no other vehicles approaching in the first 100 feet of that block. This means that for the first block, vehicles will only travel 340 feet to the next intersection in a block that is 440 feet long. For an average vehicle[1] travelling at 40mph, this provides a minimum margin of safety when considering an emergency braking deceleration rate of 14 ft/sec^2 and fast acceleration rate of 4 ft/sec^2. At the beginning of the simulation, all cars at the front of the block queues will be able to enter the simulation. The next cars in the queues will have to wait a few seconds for the car in front of them to move forward before entering the street. As the simulation progresses, some cars will have to wait for approaching traffic to pass before they can enter the street.

About 96 days of cpu time using 166MHz Pentium PCs were required to generate the results reported here. The work was distributed over eight PCs. Some PCs ran the simulations 24 hours/day and some ran the simulations only at night. This allowed the simulations to complete in about 3 weeks time.

[1] Acceleration and deceleration were timed for a Honda Civic and a Toyota 4Runner.

4. Traffic Light Control with SARSA

The SARSA (Sutton, 1996) algorithm was applied to the traffic light control problem using replace traces (Singh and Sutton, 1996) and greedy action selection. If the greedy action is not unique, one is chosen randomly. For most representations, the traffic controller is trained using experience at a single intersection with the four lanes of traffic leading into it. There are no other intersections present in the simulation during this type of training. A *lane*, relative to an intersection, is the block-long (440 foot) stretch of road approaching the intersection. The number of vehicles placed in the north-south and east-west lanes varies between zero and 50 and is chosen systematically. The vehicles' destinations are chosen randomly. After the vehicles are added to each lane, the vehicles' positions and speeds within each lane are randomly determined to the degree that all vehicles fit within their lane and are properly spaced to prevent collisions. In this way, a large variety of initial states are visited.

SARSA was implemented using a discrete state-action space to represent the states and actions. Unique Q values were associated with each discrete state-action. The current state is characterized for SARSA by the number and positions of vehicles in the north, south, east, and west lanes approaching the intersection and by the elapsed time since the last light cycle change. The action for a given state sets the color of the north-south traffic lights to red or green, and indirectly sets the color of the east-west lights, which are always set to the opposite color of the north-south lights (after cycling through the required light phases). These features were quantized and combined in four ways and the performance with the four resulting state representations was compared.

The first representation, called the vehicle *count* representation, is formed by summing the number of vehicles in the two north-south lanes and the vehicles in the two east-west lanes. These two sums are quantized into 10 partitions with values of 0, 1–5, 6–10, 11–15, ..., 36–40, and 41+. The pairs of partitions of the two sums represent 100 possible two-dimensional states. When combined with the two actions, the state-action space consists of 200 discrete states.

The second representation, called the *fixed-distance* representation, retains an indication of the relative distance of vehicles from the intersection. The intersection's lanes are divided into 110-foot intervals, forming four partitions on each of the four lanes. The presence or absence of vehicles in each partition

is determined and recorded by setting an "occupied" bit for each partition. The occupied bits for the first 110-foot partition of the north or south lanes are combined with a Boolean "or" operation, as are the north and south occupied bits for the other three corresponding partition pairs. This same process forms the representation of vehicles in the east-west lanes. Thus, the input to the learning agent with this representation is a 256-component vector with a single nonzero component. When combined with a binary value indicating the color of the north-south lights, the state-action space is represented by a 512-component binary vector with a single nonzero component.

The third representation, called the *variable-distance* representation, partitions each lane like the fixed-distance representation, but the divisions are at unequal distances from the intersection. Partition boundaries are at 50, 110, 220 and 440 feet from the intersection, making four partitions. The input to the learning agent and the state-action space with this representation consists of 256 and 512 components respectively, just like the fixed-distance representation.

The fourth representation, called the *count/duration* representation is similar to the count representation and adds the current north-south light color to the state representation and minimum light duration in seconds to the action representation. Eight counting partitions for the number of vehicles in the north-south and east west lanes were quantized for values of 0, 1–9, 10–19, 20–29, 30–34, 35–39, 40–44, and 45+. Eight light duration partitions were quantized for values of 0-4, 5–9, 10–14, 15–19, 20–24, 25–29, 30–39, and 40+ seconds. The input to the learning agent with this representation consists of a 128 component vector with one nonzero component (8 (north-south count) x 8 (east-west count) x 2 (current north-south color (0=red, 1=green and the east-west color is set to the opposite color))). The action space is represented by a 16-component vector with one nonzero component (8 (minimum light duration) x 2 (new light color)). The state-action space is represented by a 2048-component vector with one nonzero component.

For the first three representations, testing on one intersection was found to be as effective as training on any one corner, edge or middle intersection in a 4 x 4 network and more effective than training all intersections individually with their own values and eligibility traces in a 4 x 4 network. When training the fourth representation, the count/duration representation, on a single intersection with a maximum

of 50 cars per direction, the simulation does not run long enough to benefit from the longer light times and the controller performance is reduced. This representation was trained with 100, 500 and 800 vehicle loads using the full 4 x 4 traffic network. During training, each intersection had its own set of eligibility traces while a set of common state-action values was shared by all intersections. While the count/duration representation is not trained using the same method as the first three representations, the testing results can be used to compare the representations since training on a full network does not benefit the first three representations.

Table 1: SARSA Update Methods

time t	**Actual Light Colors**		**Interpreted Light Colors**		**Update Every State**	**Update Only Defined States**
	N-S	**E-W**	**N-S**	**E-W**		
1	G	R	G	R	$q(1,G) = r + q(2,G)$	$q(1,G) = r + q(2,G)$
2	G	R	G	R	$q(2,G) = r + q(3,R)$	$q(2,G) = r + q(3,R)$
3	Y	R	R	G	$q(3,R) = r + q(4,R)$	no update
4	Y	R	R	G	$q(4,R) = r + q(5,R)$	no update
5	R	R	R	G	$q(5,R) = r + q(6,R)$	no update
6	R	G	R	G	$q(6,R) = r + q(7,R)$	$q(6,R) = r + q(7,R)$
7	R	G	R	G	$q(7,R) = r + q(8,R)$	$q(7,R) = r + q(8,R)$
8	R	Y	R	G	$q(8,R) = r + q(9,R)$	no update
9	R	Y	R	G	$q(9,R) = r + q(10,R)$	no update
10	R	R	R	G	$q(10,R) = r + q(11,G)$	no update
11	G	R	G	R	$q(11,G) = r + q(12,G)$	$q(11,G) = r + q(12,R)$
12	G	R	G	R	$q(12,G) = r + q(13,a_{13})$	$q(12,G) = r + q(13,a_{13})$

When the lights cycle through the yellow/red and all/red phases the state of the environment is undefined. Two different update methods were used to handle undefined states. The update formulas are of the form: $q(s_t, a_t) = r_{t+1} + q(s_{t+1}, a_{t+1})$. The first method calculates updates during every time step even when the state is undefined. When updates are performed during every time step, the north-south light color is interpreted as being red, and the east-west light color is interpreted as being green during undefined states. The second method calculates updates only when the current state is defined.

If the current state is free of vehicles, no learning takes place since the intersection is in a goal state. When the lights cycle through a yellow or all-red phase, the environment state is not defined. Learning should only occur during defined states where one travel direction has a green light and the other direction has a red light. Two methods were used to handle updating action values when the simulator is in an undefined state. These methods are illustrated in Table 1. When SARSA updates are calculated at every time step, each state-action pair is updated based on the interpreted light color of the N-S (north-south) direction. If the north-south light color is not green, the north-south light

color is interpreted as being red. The questionable aspect of this is that the yellow/red and all-red states are always interpreted as the E-W (east-west) direction being green. This should be changed so that the direction that was green most recently is interpreted as being green during the yellow/red and all-red states. When SARSA updates are calculated only during valid defined states, not all state-action pairs are updated. For example at time, t_2, the state-action pair $q(2, G)$ is updated using the $q(3, R)$ state-action value. But at time, t_3, no update occurs and the $q(3, R)$ state-action value is not updated during this sequence. State-action, $q(3, R)$, will have to wait to be updated during another sequence of state-action sequences. Updating at every time step results in better performance for the fixed-distance, variable-distance and count representations. For the count/duration representation, better results were produced when updates were performed only during defined states. The results reported later in this paper are based on the update method that produces the best results for each representation.

Four performance measures were calculated for all tested strategies: the total number of simulation steps required for all vehicles to reach their destinations (the final goal state); average vehicle travel time; the total number of stops made by all vehicles; and the average vehicle wait time (how many seconds a vehicle has zero velocity). Tests were performed by duplicating the current state of the SARSA controller, trained on one intersection, at every intersection of a traffic network with 4 x 4 intersections. The performance measures were calculated on this test network after every 10 or 20 learning trials.

The test suite used to obtain the results in the next section consisted of 90 runs for each representation. Each run had a fixed learning rate, α, and eligibility trace decay rate, λ. The values for α ranged from 0.1 to 0.9 and λ ranged from 0.0 to 0.9. Values equal to 1.0 for α or λ caused floating point overflows. After some experimentation, the number of training trials per run was chosen to be 4,000. A trial consisted of inserting cars into the network and running the simulation until all cars reached their destination. Trials were limited to 1,200 time steps for 100 and 500 car tests and 2,400 time steps for 1,000 car tests. After 10 to 20 training trials, three tests were run consisting of 100, 500 and 1,000 cars. The *combined* metric was formed by summing the results of the 100, 500 and 1,000 car tests at each testing trial during a run. For example, if the number of stops for testing trial 20 is 200, 1100 and 3000 for 100, 500 and 1,000 cars respectively, the combined metric is 4300 stops at trial 20. This process is repeated

to calculate the other three combined metrics. The weighted average of the combined metric is then calculated and searched to find the best performance when considering all three traffic loads. The Q values were initialized to zero before the start of each run. Training and testing used about 2 days of wall clock time on a 166MHz Pentium processor for the first three representations and about 90 days for the count/duration representation.

Two types of reinforcements were tested for each representation. The first method used a reinforcement of -1 for each discrete time step. The second method used a reinforcement that varied from -5 to -1 depending on sensor activations as follows. There are four sensors near an intersection, one in each lane, that are able to detect moving or stopped traffic. A sensor that is activated in a lane with a green light will detect moving traffic and a sensor that is activated in a lane with a red light will detect stopped traffic. The reinforcement is initially -3. If one of the lane sensors corresponding to a green light (travelling traffic) is activated one or more times during the previous second, the reinforcement is incremented by 1 to reward the controller for moving traffic through the intersection. If both green light sensors are activated, the reinforcement is incremented by 2. If a lane sensor corresponding to a red light (stopped traffic) has been activated during the previous second, the reinforcement is decremented by 1 to punish the controller for stopping traffic. If both red light sensors are activated, the reinforcement is decremented by 2. This should help the controller learn faster that stopping traffic in one direction while there is no traffic in the other direction is not a good choice. If there is an equal amount of traffic moving and waiting at an intersection, a neutral reward of -3 is assigned. No discount was used, since the trials are finite in length.

The two reinforcement schemes are compared in Figure 5 for three SARSA representations by plotting the number of simulation steps to reach the final goal versus trials using $\alpha = 0.3$ and $\lambda = 0.1$. The results from the fixed-distance representation are similar to the variable-distance representation results and are not included. The varying reinforcements improve the performance and reduce the variation in the results for the fixed-distance and variable-distance representations, while the reverse holds for the count/duration representation. There is no significant difference for the count representation. For the primitive representations, varying the reinforcement helps to evaluate the actions more precisely and to

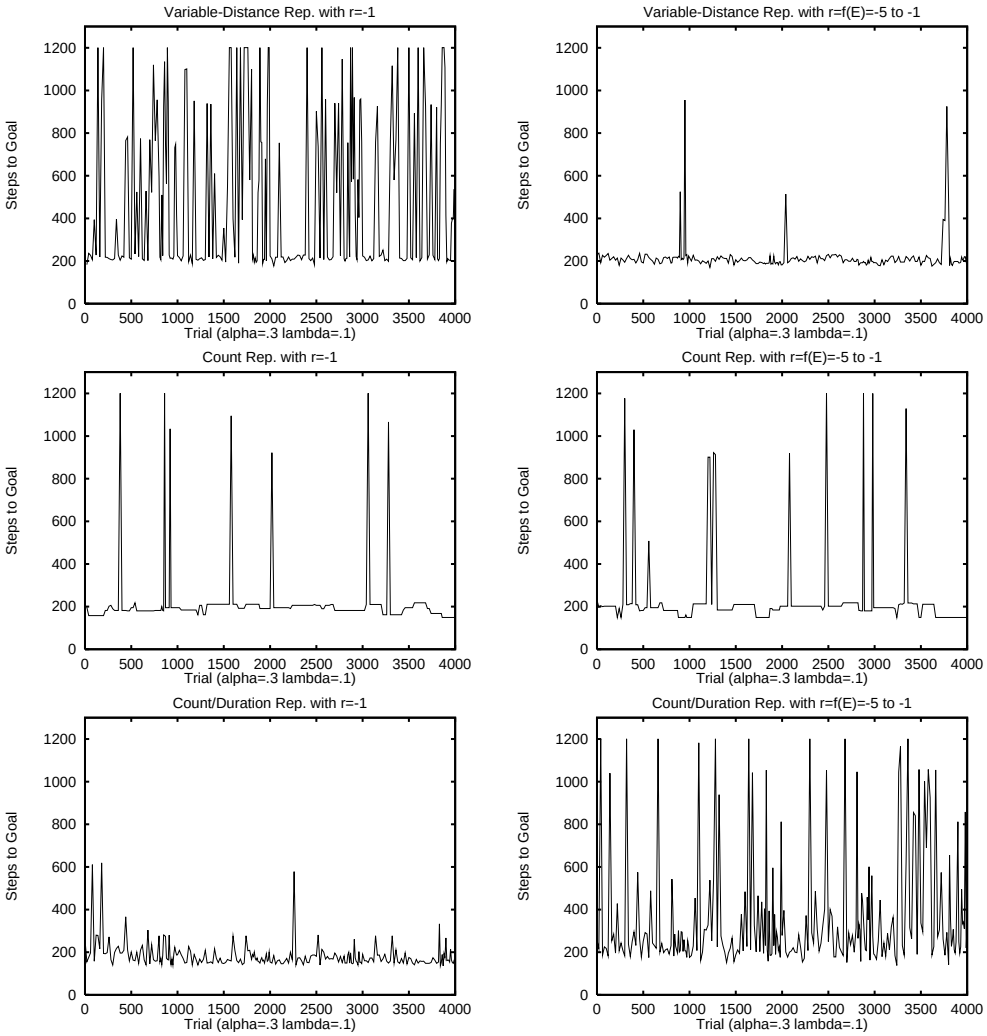

Figure 5: The effect of two reinforcement systems are compared for three SARSA representations based on the total number of steps for 100 cars to reach their destinations. A varying reinforcement based on the environment, $r = f(E) = -3 +$ green-sensor hits $-$ red-sensor hits, dramatically reduces the variation in the primitive variable-distance representation and has little effect on the count representation. The results for the fixed-distance representation are similar to the variable-distance representation. A constant reinforcement of -1 is the best for the count/duration representation.

reduce the variation during training considerably. For the remaining comparisons, varying reinforcements are used for the primitive representations and a constant reinforcement is used for the count/duration representation.

Figure 5 also reveals two apparent problems. First, it appears that no learning takes place or occurs during the first 20 to 40 trials. Second, with the large performance variation between neighboring test points, how can the performance of different representations and learning parameters be fairly characterized? Using the minimum-valued test point would reflect better performance than might actually be achievable under varying traffic loads. A learning trial is characterized by finding the minimum-valued test point in a trial based on the weighted average of the test points. The weighted average for the kth test point is calculated as follows:

$$w(k) = \left. \frac{\sum_{i=1}^{n} \frac{i}{n} y_{k-n+i-1} + \frac{n+1}{n} y_k + \sum_{j=1}^{n} \frac{n-j+1}{n} y_{k+j}}{\sum_{i=1}^{n} \frac{i}{n} + \frac{n+1}{n} + \sum_{j=1}^{n} \frac{n-j+1}{n}} \right| k-n+1-i > 0, k+j <= m$$

where w is the weighted average of the kth point, y is the actual value of point, k is the index of the point being averaged, m is the index of the last test point and n is the number of points on each side of the test point being weighted with the test point.

Figure 6 shows the weighted average along with the original reward comparisons from Figure 5. The weighted average for a point was determined by averaging it with 30 neighboring points. The effect of the two reinforcement systems and the extent of the variation for the different representations is easier to see. The variable-distance representation using varying rewards does not exhibit much variation but is pretty flat and does not appear to be learning. Learning may be occurring for the count/duratation representation with a constant reinforcement of -1 and for both of the count representations.

A 31 test point average ($n = 15$) is used for all of the SARSA representation performance comparisons. Figures 7 and 8 compare the learning progress for the SARSA representations tested in a 4 x 4 network with 100 cars, 500 cars, 1000 cars and combined loads with $\alpha = 0.1$ and $\lambda = 0.4$. The weighted average is labeled, "31 Test Point Average", in the figures. The figures also include the performance for each representation selecting actions randomly, the all-green strategy and the best fixed-duration strategy for each traffic load. The "31 Test Point Average" shows the learning progress more clearly. Learning is

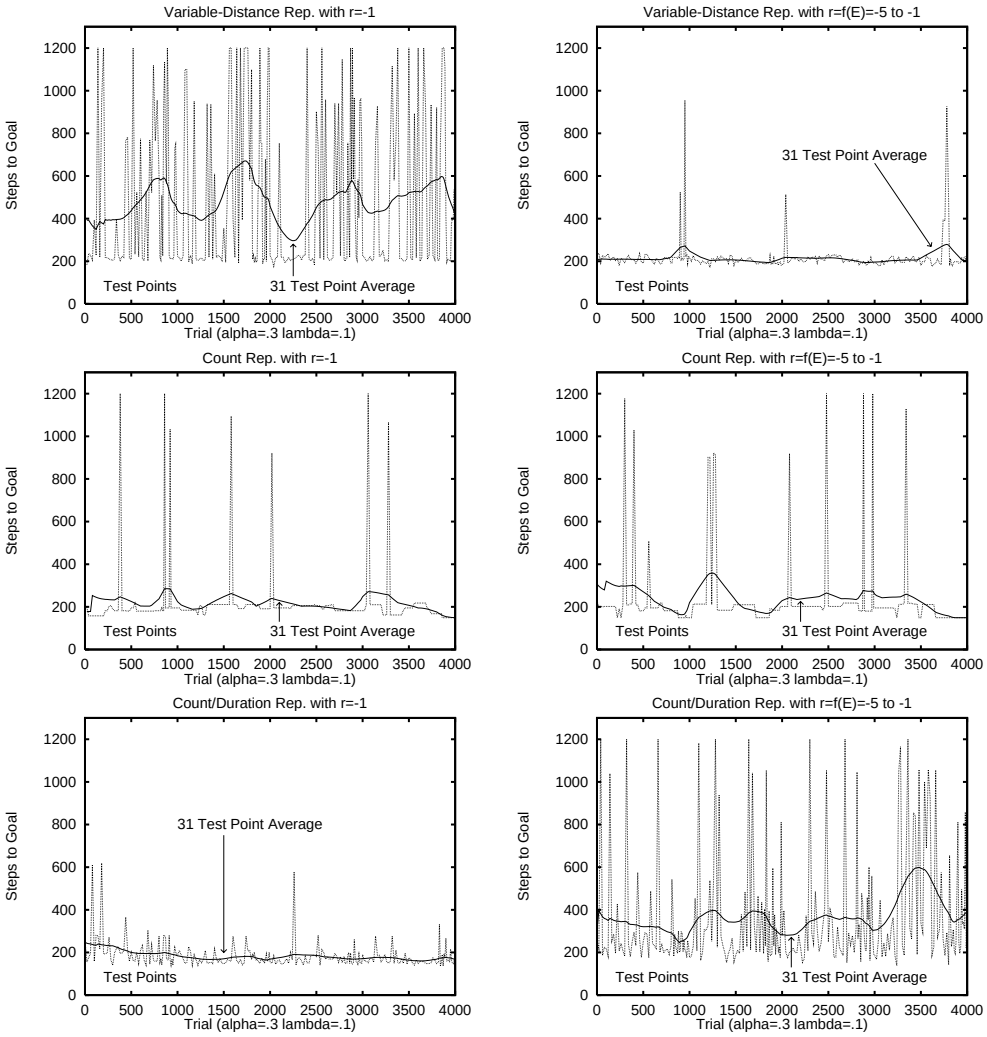

Figure 6: The results from Figure 5 are shown with the weighted average of the total number of steps for 100 cars to reach their destination superimposed. The weighted average for a point was determined by averaging 30 neighboring points with the point. The effect of the two reinforcement systems and the extent of the variation for the different representations is easier to see. Learning may be occurring for both count representations and the count/duratation representation with a constant reinforcement of -1.

more evident in the 500 car, 1000 car and combined runs. The count and count/duration representation show the most consistent learning progress followed by the fixed-distance representation. The variable-distance representation experiences large oscillations in performance about every 1000 trials with the final trials being worse than the initial trials for the 1000 car and combined runs. Only a few of the representations perform better than the fixed-duration representation. These learning parameters were selected because they illustrate the learning progress more clearly than other learning parameters. Other learning parameters do not perform as well as selecting actions randomly and other learning parameters will be shown to perform better than the fixed-duration strategy.

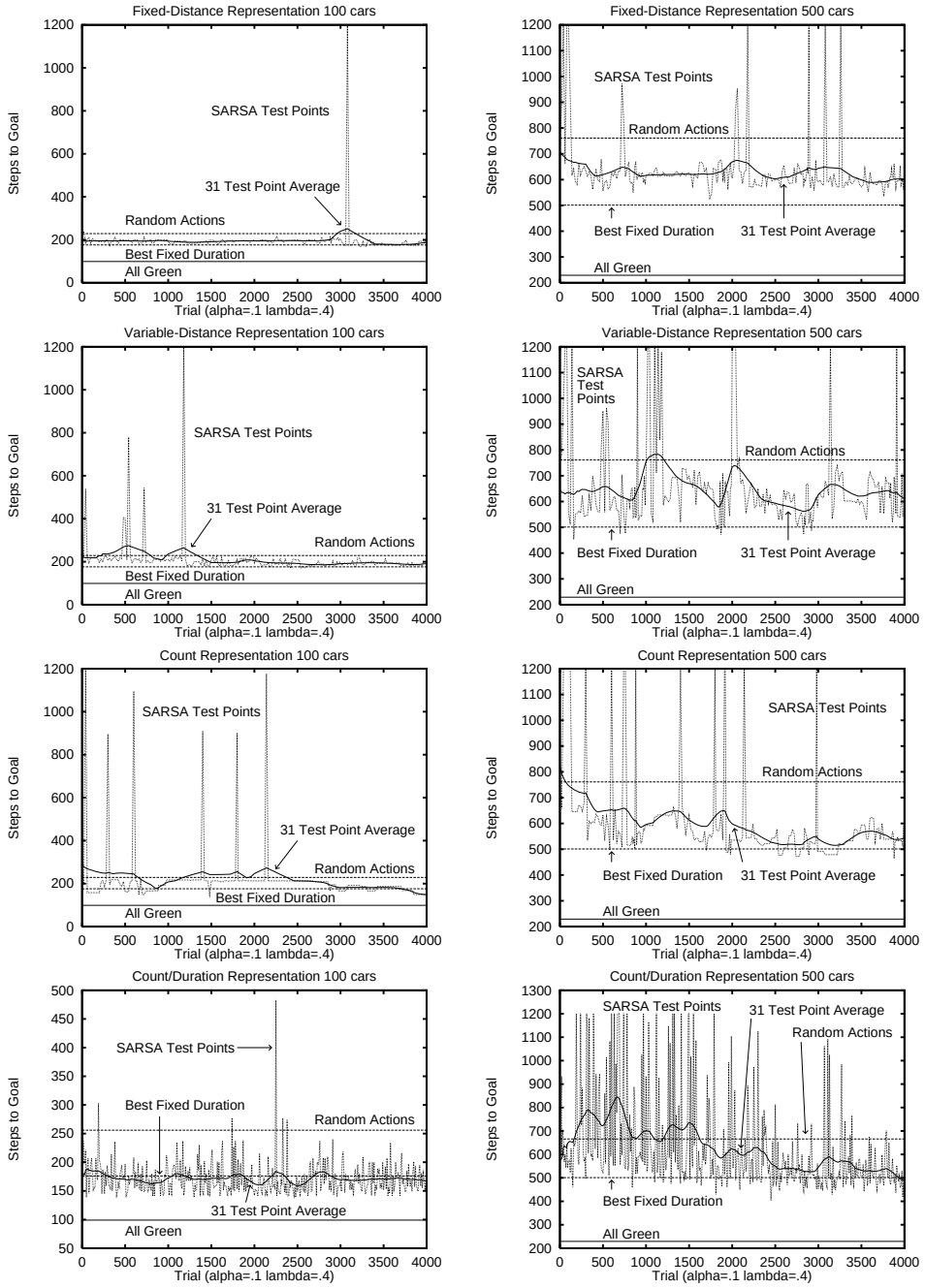

Figure 7: Comparison of learning progress for the SARSA representations tested in a 4 x 4 test network is shown with 100 and 500 cars. The "31 Test Point Average" helps show the learning tendency of each representation. Learning is more apparent in the 500 car load. The SARSA reinforcement is $r = -1$ for the count/duration representation and $r = f(E)$ for the count representation.

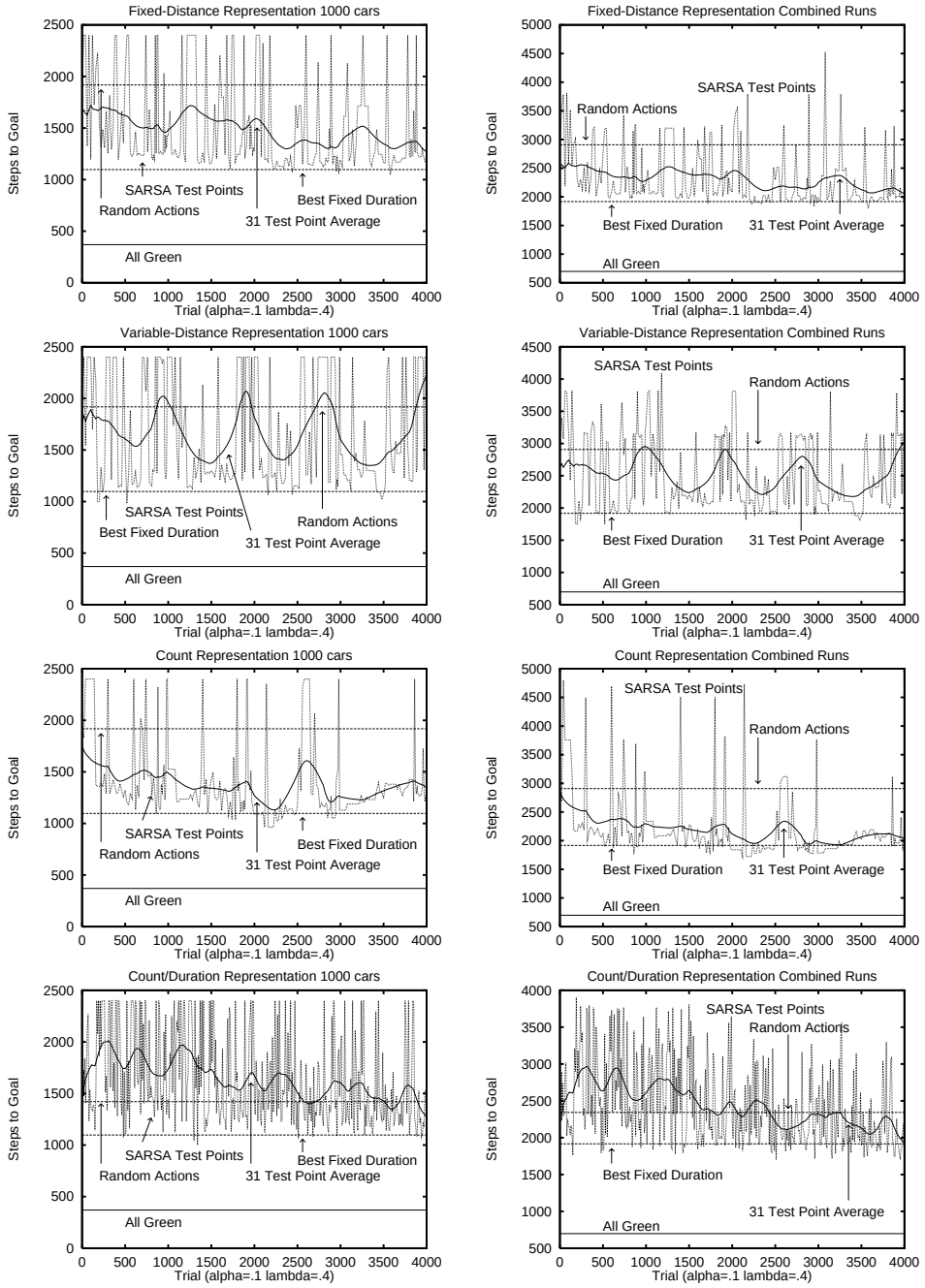

Figure 8: Comparison of learning progress for the SARSA representations tested in a 4 x 4 test network is shown with 1000 cars and combined runs. Learning is easily seen except for the variable-distance representation which experiences large oscillations every 1000 tests. The SARSA reinforcement is $r = -1$ for the count/duration representation and $r = f(E)$ for the count representation.

5. Results

The fixed-duration strategy was tested by varying the green light duration for varying loads. Figure 9 shows the number of steps required to clear the simulator of all vehicles versus the green light timing. Table 2 summarizes the best, fixed-light timing for all metrics.

To reduce the total number of steps for all cars to reach their destination, the optimum green light duration occurs at 17 seconds for 100 cars and increases to 61 seconds for 500 cars and 112 seconds for 1,000 cars. Different fixed-light timing ranges are needed to generate the best performance for the other metrics. This shows that a simple fixed-duration strategy will not work for varying traffic loads. The best fixed-duration value for each load is shown in subsequent comparison graphs.

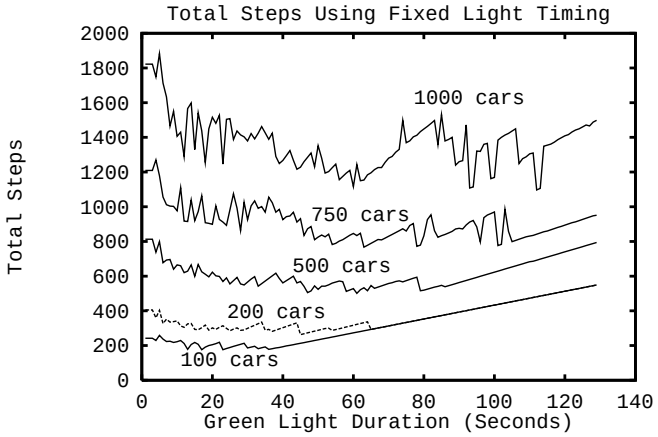

Figure 9: A comparison of fixed-duration light timing settings in a 4x4 test network shows that different timings provide optimal performance at different traffic loads.

For the 100, 200 and 500 car loads it is clear that after a certain green-light duration, the total number of steps required to complete the simulation increases linearly as the green light duration increases. An interesting "quantum" like phenomena may be occurring in the 750 and 1000 car traffic loads. Consider the 750 car load. At about 65 seconds through 75 seconds green light duration, the total number of steps increases mostly linearly. At about the 78 second timing mark, there is a sudden drop in the total number of steps. These valleys may be occurring due to the traffic lights synchronizing well with the traffic flow. These "quantum" like jumps may also be present in the lighter traffic loads but are not as obvious.

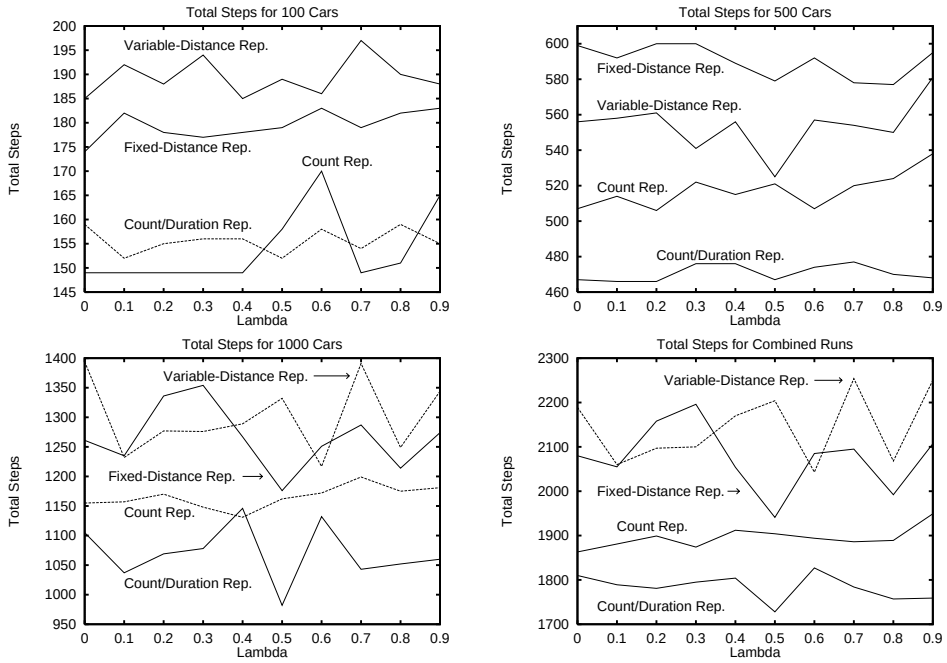

Figure 10: Comparison of SARSA representations using the best weighted average of total steps for all cars to reach their destination versus eligibility trace decay λ in a 4 x 4 test network. The best value for α is used for each λ and SARSA representation combination. The SARSA reinforcement is -1 for the count/duration representation and variable for all other SARSA representations.

Car Count	**Total Steps**	**Total Stops**	**Avg Travel Time per Car**	**Avg Wait Time per Car**
100	17 secs	45 secs	47 secs	7 secs
500	61 secs	117 secs	55 secs	17 secs
1000	112 secs	128 secs	93 secs	18 secs
Combined	60 secs	128 secs	53 secs	12 secs

Table 2: Best Fixed-Duration Settings for each Metric and Traffic Load

The performance of the SARSA algorithm was tested with the four representations described above. For each representation, the best values of α and λ were selected by searching for the best weighted average of the metric under test. Figure 10 shows the best weighted average of total steps for all cars to reach their destination versus λ. The control strategies in best to worst order are:

1. SARSA count/duration representation;
2. SARSA count representation;
3. SARSA fixed-distance representation;
4. SARSA variable-distance representation.

Since there would be less variation in the location of vehicles at high traffic loads, it was expected the fixed-distance and variable-distance representations would perform better under light loads. However, the variable-distance representation performs better for medium loads and poorly overall. The performance difference between the variable-distance and fixed-distance representations shows that knowing vehicle locations more precisely as the vehicles get closer to the controlling intersection, reduces, rather than increases, the controller performance. The count representation performed at a relatively moderate level, except for light loads when it was the best. The count/duration representation performed best under all except light loads. The best overall λ occurred at 0.5 or 0.6.

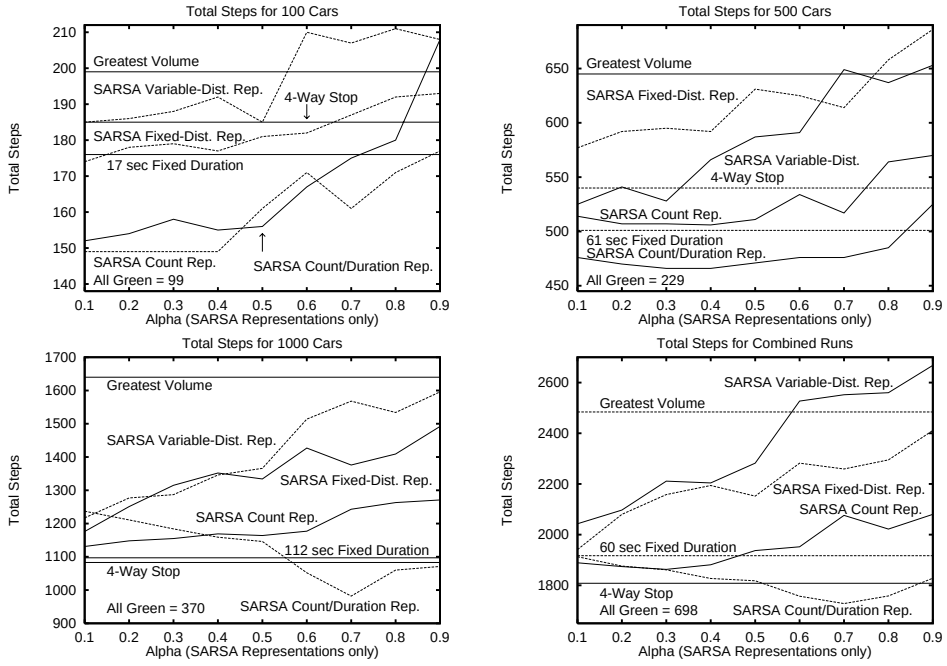

Figure 11: Strategy comparison using best weighted average of total steps for all cars to reach their destination versus learning rate α in a 4 x 4 network. There is not as much variation between α values as compared to λ values in Figure 10. The best value for λ is used and varies for each α within each SARSA representation.

Figure 11 shows the best weighted average of total steps for all cars to reach their destination versus α for all strategies. Some of the strategies are not shown on the graphs so that the differences between the remaining strategies are easier to observe. When a strategy has not been graphed, its performance is included on the graph as a label. All SARSA representation are included on all graphs. The control strategies in best-to-worst order for the total number of steps for all cars to reach their destination under

heavy traffic loads (1000 cars) was found to be:

1. all green lights;
2. SARSA count/duration representation;
3. four-way stop;
4. 112 second fixed-duration;
5. SARSA count representation;
6. SARSA fixed-distance representation;
7. SARSA variable-distance representation;
8. greatest-volume strategy.

For the primitive SARSA representations, as α was increased, the performance generally decreased. For the count/duration representation, the best α started at 0.1 for light loads and increased to 0.7 for higher loads. The best values for α to reduce the total simulation steps range from 0.5 through 0.9.

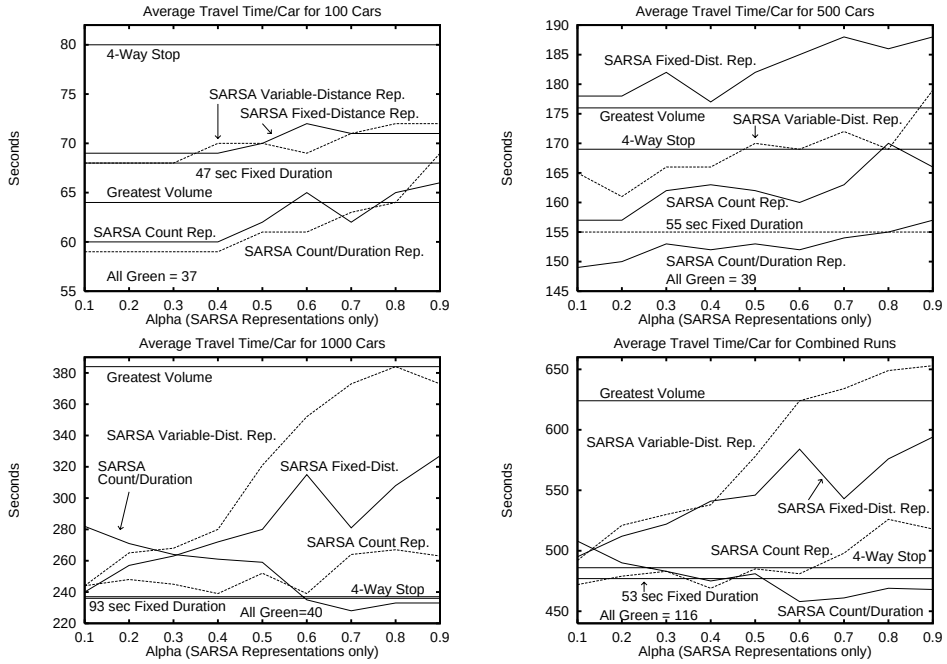

Figure 12: Strategy comparison using the best weighted average of travel time/car for cars to reach their destination in a 4 x 4 network. The best value for λ is used and varies for each α within each SARSA representation.

Figure 12 compares the weighted average of travel time per vehicle versus α for all strategies. The control strategies in best-to-worst order for the average of travel time per vehicle under heavy traffic loads (1000 cars) was found to be:

1. all green lights;
2. SARSA count/duration representation;
3. 93 second fixed-duration;
4. four-way stop;
5. SARSA count representation;
6. SARSA fixed-distance representation;
7. SARSA variable-distance representation;
8. greatest-volume strategy.

There is only a 16-second variation for the best results for all of the SARSA representations under heavy load. Any SARSA representation would be acceptable for reducing the average travel time/car since the differences in performance would probably not be noticed by the average individual. The greatest-volume strategy performs better than the fixed-distance representation for medium loads. Timed lights, four-way stops and the greatest-volume strategies performed better as the traffic loads increased. The greatest-volume strategy performed better than the fixed-distance representation under light and medium loads and better than the variable-distance strategy under light loads. The count/duration representation was consistently the best strategy for all traffic loads. The average travel time/car increased as α was increased for all SARSA representations except for the count/duration at heavy and combined loads. The best α for the count/duration representation at heavy and combined loads was 0.7 and 0.6 respectively.

Figure 13 shows that the control strategies in best-to-worst order for the weighted average of total stops per vehicle under heavy traffic loads (1000 cars) are:

1. all green lights;
2. 128 second fixed-duration;
3. SARSA count/duration representation;
4. SARSA count representation;
5. SARSA variable-distance representation;
6. SARSA fixed-distance representation;
7. greatest-volume strategy;
8. four-way stop.

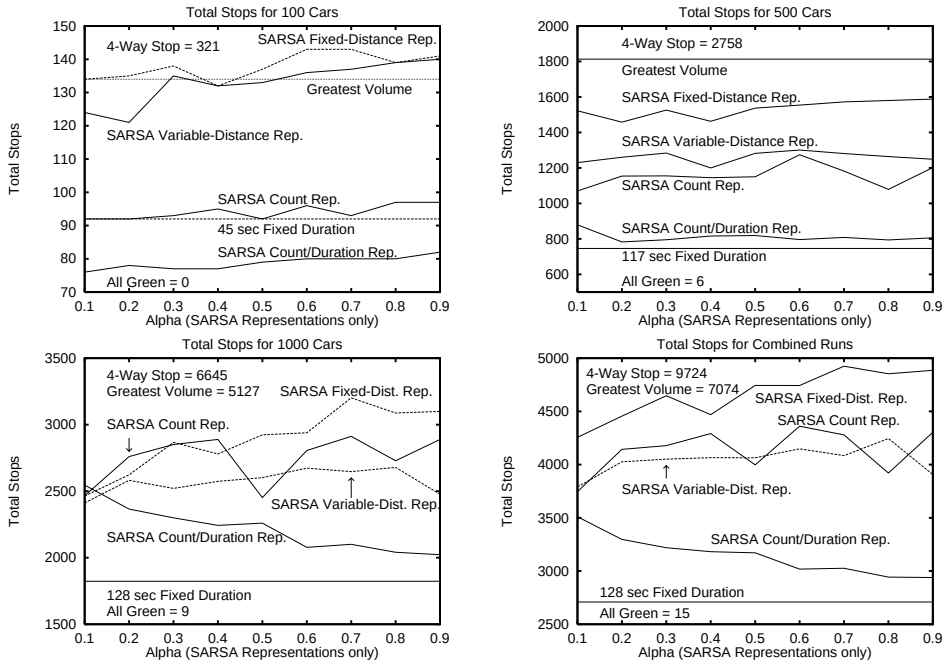

Figure 13: Strategy comparison using the best weighted average of total stops for all vehicles en route to their destination in a 4 x 4 network. The best value for λ is used and varies for each α within each SARSA representation.

As expected, the four-way stop experienced more stops than any other strategy. It was thought that the varying reinforcement for the three primitive SARSA representations which receive higher reinforcements for moving traffic along and experience greater punishment for stopped traffic would result in fewer stops, but this did not happen. The fixed-duration timing strategy performed better than the SARSA representations as the traffic loads increased. This shows that the SARSA representations and reinforcements are not optimized to reduce the number of stops a vehicle makes. However, there is only about a difference of 0.5 stops per car between the fixed-duration strategies and SARSA representations which would not be noticed by the average driver. The total number of stops increased as α was increased for all SARSA representations except for the count/duration at medium, heavy and combined loads. The best α for the count/duration representation at medium, heavy and combined loads was 0.2, 0.9 and 0.9 respectively.

Figure 14 shows that the control strategies in best-to-worst order for the average wait time per vehicle under heavy traffic loads (1000 cars) are:

1. all green lights;

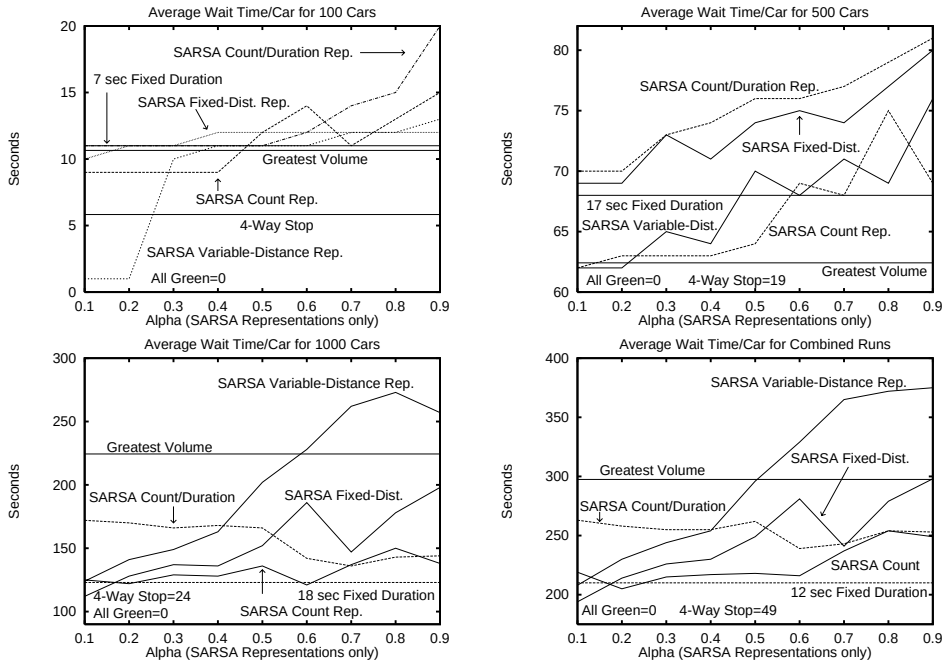

Figure 14: Strategy comparison using the best weighted average of wait time/car en route to their destination in a 4 x 4 network. The best value for λ is used and varies for each α within each SARSA representation.

2. four-way stop;
3. SARSA fixed-distance representation;
4. SARSA count representation;
5. 18 second fixed-duration;
6. SARSA variable-distance representation;
7. SARSA count/duration representation;
8. greatest-volume strategy.

For this metric, the relative order of performance for the SARSA representations has changed drastically compared to the other metrics. Here, the fixed-distance representation performs the best overall. The count/duration representation, instead of consistently being the best representation, is always the worst. The fixed-distance and variable-distance representations, which for the previous metrics were the worst, are among the best especially as the traffic loads increase. It could be that the varying reinforcement schemes are helping reduce the average wait time except that tests with the reinforcement schemes reversed show the same ordering of control strategy performance for the high and combined loads and similar ordering for the low and medium loads. The four-way strategy which performed average to worst

for the other three metrics is better than all of the other strategies except for the all-green strategy. The four-way stop strategy was able to reduce the amount of time that vehicles were completely stopped by continually moving traffic slowly through the environment. The average wait time/car increased as α was increased for all SARSA representations except for the count/duration at heavy and combined loads. The best α for the count/duration representation at heavy and combined loads was 0.7 and 0.6 respectively.

Figure 15 and Figure 16 compares the performance for different α and λ values for the four SARSA representations for combined traffic loads. The variable-distance, fixed-distance and count representations perform best with small α and the count/duration representation performs best at $\alpha = 0.7$. The value of λ is not as critical in any representation. The variable-distance, fixed-distance and count representations generally form smoother surfaces than the count/duration representation. This may be an indication that the count/duration representation which is more complex than the other representations may not have reached their optimal values and could benefit from more training.

Figure 17 shows the average green light duration versus α. The order of the strategies from shortest to longest average green light duration was:

1. SARSA fixed-distance representation;
2. SARSA variable-distance representation;
3. SARSA count representation;
4. greatest-volume strategy.
5. SARSA count/duration representation;

For the fixed-distance and the variable-distance representations, the green light duration increased slightly as the traffic load increased. This was expected but it was thought the increases would have been larger than 1 to 5 seconds between loads. For the count and count/duration representations, the green light duration increased from light to medium traffic loads and decreased slightly from medium to heavy traffic loads. For all except the count/duration representation, the green light duration appears to be independent of α. For the count/duration representation, the green light duration increases as α increases. This may be another indication that the count/duration representation might benefit from further training. Since the larger α values have longer green light durations, further training with the

smaller learning parameters might bring the green light duration up to the same levels as the larger α values. The green light duration for the count, fixed-distance and variable-distance are all within 5 seconds of each other. The count/duration green light duration is 15 to 17 seconds longer than the other SARSA representations. The longer green light times for the count/duration representation help to make it the best representation for reducing the total simulation time, average travel time and total vehicle stops.

Figure 18 shows the percentage of state-action spaces visited during training. The count representation visited 99% of the state-action pairs (all except the two goal state-action pairs) in about 900 trials. The variable-distance and fixed-distance representation graphs are very similar in shape but scale differently. The variable-distance representation consistently visited about 50% more state-action pairs than the fixed-distance representation. The variable-distance and fixed-distance representations visited about 75% and 52% of the state-space, respectively, after 3200 trials. When training on one intersection, the cars will crowd forward towards the intersection when waiting for a red light to turn green. Since the variable-distance representation has more partitions closer to the intersection, it can visit more states than the fixed-distance representation using the same training method. The systematic selection of the number of vehicles used during training is visible at the 1000, 2000 and 3000 trial marks. The count/duration representation fell between the variable-distance and fixed-distance representations visiting about 58% of the state-action space at the end of the run.

To see if artificial off-line training on one intersection is a good idea, five other on-line training methods based on the count representation with $\alpha = 0.1$ and $\lambda = 0.4$ were tested. Two tests were performed with training occurring on all 16 intersections in a 4x4 network: one with each intersection having its own eligibility traces and action values; and another with each intersection having its own eligibility traces but sharing action values. Three more learning tests were performed on the full environment but learning only occurred on a corner intersection, non-corner edge intersection or an interior/center intersection.

The best weighted average results from each training method are shown in Figure 19. As expected, the learning strategies in best-to-worst order are:

1. learning on all intersections in a 4x4 network with all intersections sharing and updating the same state-action values;
2. artificial learning on one intersection;
3. learning on a center intersection in a 4x4 network;
4. learning on all intersections in a 4x4 network with each intersection having its own state-action values;
5. learning on an edge intersection in a 4x4 network;
6. learning on a corner intersection in a 4x4 network.

The performance of the training method seems to be directly related to the amount of traffic each method experiences. The more traffic that the training method receives, the better the performance from that training method. The corner training would be the worst since a corner would not be expected to receive as much traffic as the interior intersections and would perform poorly when interior intersections used a corner policy. Training on an edge intersection performed better than corner training since an edge should experience more traffic than a corner intersection. Training on an artificial intersection and training on all intersections that share action values performed the best and very close to each other. For light loads the training methods were nearly identical. Training becomes more critical as the load increases.

The ability of the control policy to generalize is an important aspect of any controller. For this report all testing was done with the same random seed. To see how well the controllers generalize, the best policy and an average policy found during training were both tested using 100 other random seeds. The variable-distance representation was used with $\alpha = 0.7$ and $\lambda = 0.0$. The results are shown in Figure 20. The first data point uses the same random seed that was used for all other testing in this report. The remaining data-points were tested using random seeds that were generated randomly.

While the best policy performed better than the average policy overall, there were several tests where the average policy out-performed the best policy. Testing should include results based on 10 to 20 different random seeds to measure the performance of the controller. This may or may not indicate a problem with the control strategy. Using a random seed may not be the best method to train and test the controller. Unusual traffic densities might result that would not occur naturally. An appropriate probability distribution should be determined and used for future testing and training.

The best results based on weighted averages and the best learning parameters of all strategies are summarized in Tables 3 and 4, respectively. The most noticeable observation from Table 3 is that SARSA performs best when considering the total number of steps to goal, average wait and average travel time. Since a negative reinforcement is used for each time step, the resulting control policy will try to reach the goal as quickly as possible. As a result, all metrics, except the number of stops, are optimized as much as possible. Table 4 shows the best parameters for each strategy. For the fixed-distance, variable-distance and count representations, small values for α produce the best results for most metrics. For the count/duration representation, the best values for α increase as the traffic load increases. There is no consistency concerning the best values for λ. They vary widely within each representation, metric and traffic load.

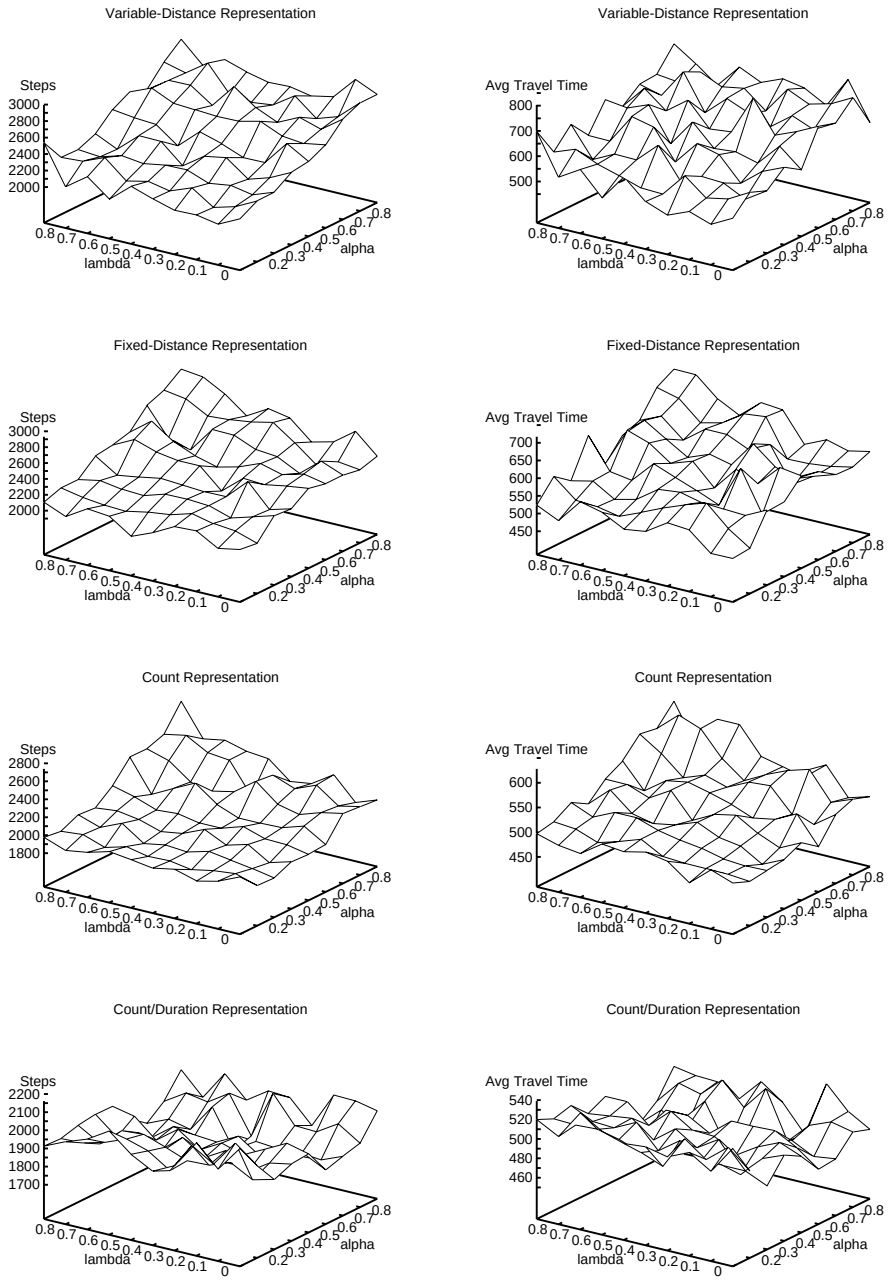

Figure 15: A comparison of the best weighted average for Total Steps and Average Travel Time per vehicle in seconds versus α and λ parameters for combined runs is shown for each SARSA representation. The variable-distance, fixed-distance and count representations perform best with small α and the count/duration representation performs best at $\alpha = 0.7$. The value of λ is not as critical in any representation. The SARSA reinforcement for the Count/Duration representation is $r = -1$ and $r = f(E)$ for the other representations.

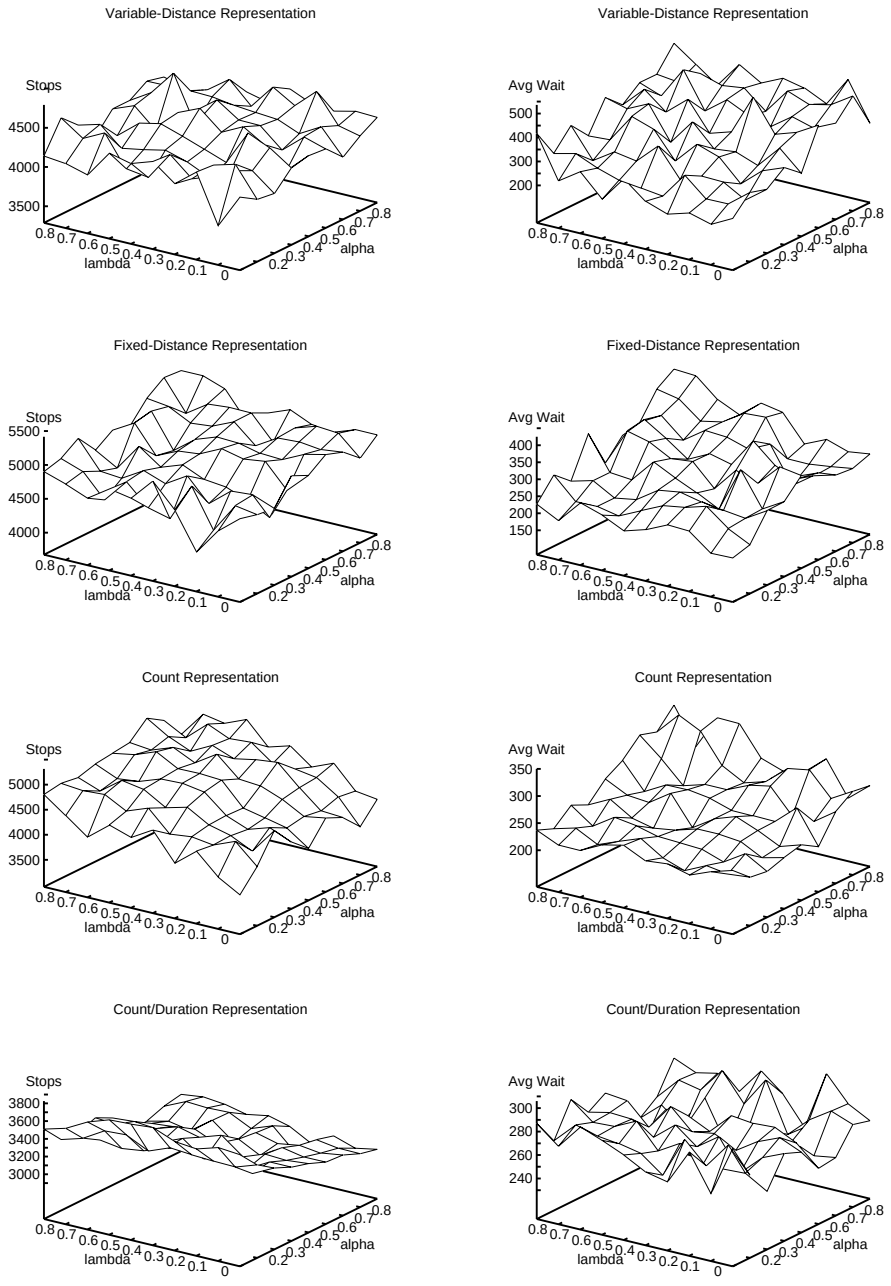

Figure 16: A comparison of the best weighted average for Total Stops and Average Wait Time per vehicle in seconds versus α and λ parameters for combined runs is shown for each SARSA representation. The variable-distance, fixed-distance and count representations perform best with small α and the count/duration representation performs best at $\alpha = 0.7$. The value of λ is not as critical in any representation. The SARSA reinforcement for the Count/Duration representation is $r = -1$ and $r = f(E)$ for the other representations.

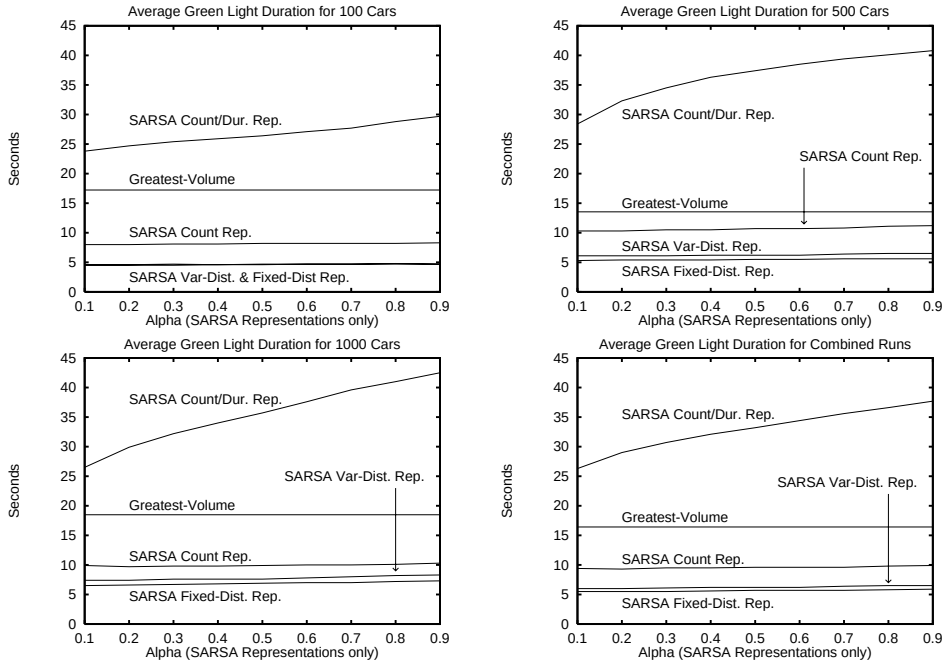

Figure 17: The average green light duration is shown for each SARSA representation versus α. The best value for λ is used and varies for each α within each SARSA representation.

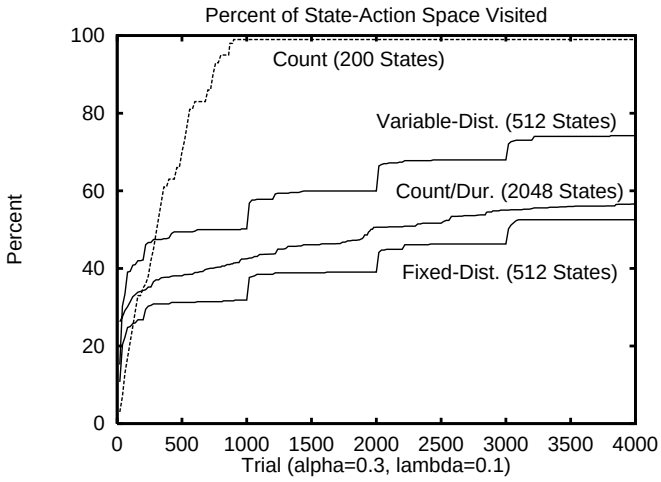

Figure 18: Percentage of state action spaces visited during learning for SARSA representations.

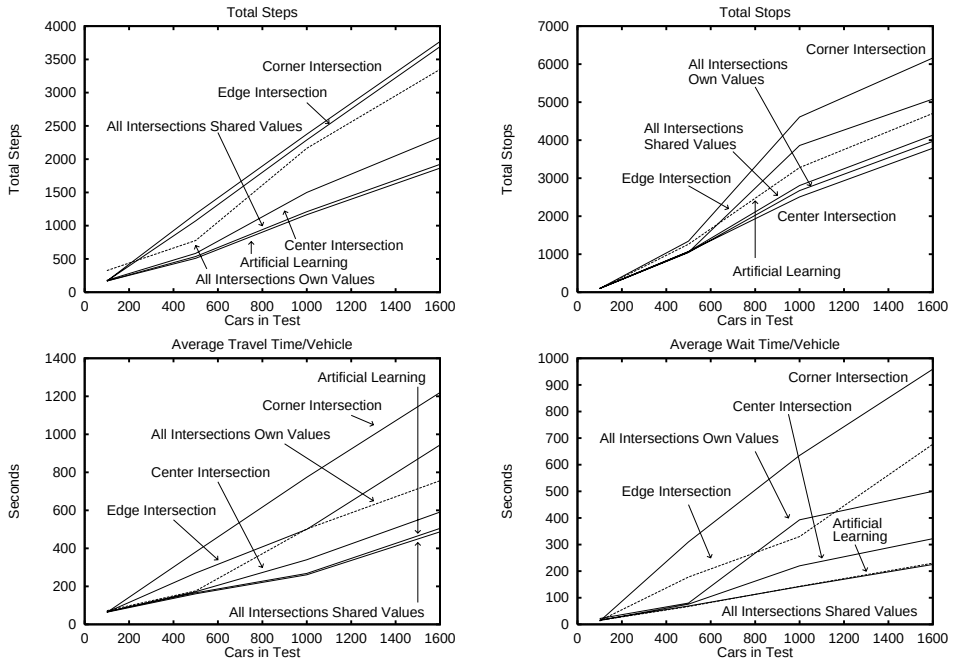

Figure 19: Learning strategy comparison using the SARSA count representation for 100, 500, 1000 and combined runs, $\alpha = 0.1$ and $\lambda = 0.4$. The training methods that experience the most traffic at the intersections where learning occurs, perform the best.

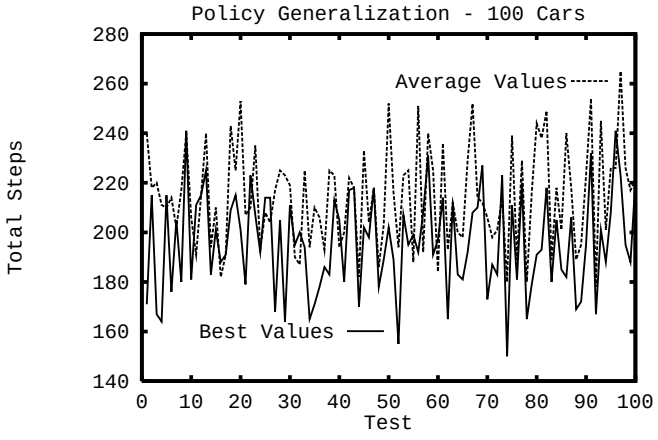

Figure 20: Ability of selected policies to generalize with different random seeds using variable-distance representation, $\alpha = 0.7$ and $\lambda = 0.0$. The best-result policy generally has better overall performance, but its performance varies more than the average-result policy.

Table 3: Best SARSA Weighted Average Result Comparisons

Strategy	**Car Count**	**Total Steps**	**Total Stops**	**Avg Travel Time per Car** (secs)	**Avg Wait Time per Car** (secs)
Random	100	229	179	80	17
Greatest-Volume	100	199	134	64	11
Variable-Distance	100	185	121	68	10
Four-Way Stop	100	185	321	80	6
Fixed-Duration	100	176	92	68	11
Fixed-Distance	100	174	132	69	10
Count/Duration	100	152	76	59	11
Count Cars	100	149	92	60	9
All-Green	100	99	0	37	0
Random	500	761	2490	230	89
Greatest-Volume	500	645	1813	176	62
Fixed-Distance	500	577	1458	177	69
Four-Way Stop	500	540	2758	169	19
Variable-Distance	500	525	1200	161	62
Count Cars	500	506	1070	157	62
Fixed-Duration	500	501	746	155	68
Count/Duration	500	466	783	149	70
All-Green	500	229	6	39	0
Random	1000	1918	5584	382	179
Greatest-Volume	1000	1640	5127	384	224
Variable-Distance	1000	1217	2412	244	124
Fixed-Distance	1000	1176	2464	240	112
Count Cars	1000	1131	2452	239	121
Fixed-Duration	1000	1097	1823	236	123
Four-Way Stop	1000	1083	6645	237	25
Count/Duration	1000	982	2023	228	136
All-Green	1000	370	9	40	0
Random	Combined	2908	8523	692	285
Greatest-Volume	Combined	2484	7074	624	297
Variable-Distance	Combined	2043	3790	492	208
Fixed-Distance	Combined	1941	4256	495	194
Fixed-Duration	Combined	1917	2709	477	210
Count Cars	Combined	1863	3746	469	205
Four-Way Stop	Combined	1808	9724	486	49
Count/Duration	Combined	1728	2939	458	239
All-Green	Combined	698	15	116	0

Table 4: Best Performance Parameters

Strategy	**Car Count**	**Total Steps**	**Total Stops**	**Avg Travel Time per Car**	**Avg Wait Time per Car**
SARSA Fixed-Distance		(α/λ)	(α/λ)	(α/λ)	(α/λ)
	100	.1/.0	.4/.1	.1-.4/.0,.1,.3-.6	.1/.5
	500	.1/.8	.2/.0	.4/.1	.1,.2/.4,.6
	1000	.1/.5	.1/.2	.1/.8	.1/.8
	Combined	.1/.5	.1/.2	.1/.8	.1/.8
SARSA Variable-Distance		(α/λ)	(α/λ)	(α/λ)	(α/λ)
	100	.1,.5/.0,.4	.2/.5	.1-.3/.0,.5.7,.8	.3/.4-.7
	500	.1/.5	.4/.9	.2/.3	.1,.2/.3,.4
	1000	.1/.6	.1/.1	.1/.6	.1/.6
	Combined	.1/.6	.1/.2	.1/.3	.1/.6
SARSA Count		(α/λ)	(α/λ)	(α/λ)	(α/λ)
	100	.1,.2,.5/.0-.4,.7	.1-.4/.0-.4	.1-.4/.1-.4,.7,.8	.0-.4/.0-.4
	500	.1/.2	.4/.1	.1/.2	.1/.8
	1000	.1/.4	.5/.0	.4,.6/.0,.1	.6/.0
	Combined	.3/.0	.1/.0	.4/.1	.2/.3
SARSA Count/ Duration		(α/λ)	(α/λ)	(α/λ)	(α/λ)
	100	.1/.1,.5	.1/.1	.1-.4/.0-.5	.1-.5/any
	500	.3,.4/.1,.2	.2/.9	.1/.2	.1,.2/.0,.2
	1000	.7/.5	.9/.3	.7/.5	.7/.5
	Combined	.7/.5	.9/.3	.6/.8	.6/.8

6. Conclusions and Future Work

For all traffic loads, SARSA learned control strategies using the count/duration representation that were better than all of the fixed strategies (except all-green) for minimizing the total traffic travel time and individual vehicle travel times. The SARSA fixed-distance and variable-distance representations were the most effective in reducing average vehicle wait time. The fixed-duration strategy was the best strategy for reducing the total number of vehicle stops. Since the average driver would not typically notice the difference in performance between any of the strategies for controlling average wait time and total number of vehicles stops, the best overall state representation for SARSA is the count/duration representation. Finding a good representation is critical to the success of the SARSA strategy. The fixed-distance, variable-distance and count representations did not take into account the current light color and were not able to achieve long green light durations which appear to improve the performance of the controller. The count/duration representation did account for the current traffic light color and achieved longer green lights by specifying a minimum green light duration. The drawback of the count/duration representation is that it takes about 40 times as long to train as the other representations.

The greatest-volume strategy was not very effective, probably because it is very sensitive to traffic fluctuations and may flip-flop between light colors excessively. This could cause traffic delays at times when the traffic volumes in the north-south and east-west lanes are nearly equal. Under these circumstances, when one or two cars travel through an intersection, there will be more cars in the stopped lanes and the controller will switch the right-of-way to the stopped lanes. Before giving the stopped lanes the right-of-way, the lights must cycle through the yellow and all-red phases first. Then, after a few cars travel through the intersection, the controller may switch the lights again. If the lights regularly flip-flop, the lights may be yellow or all-red more often than giving either direction a green light. The count and count/duration representations do not have this problem, even though they are similar to the greatest-volume strategy, because partitioning the vehicle counts dampens the effect of traffic fluctuations. The minimum green-light duration of the count/duration representation helps to reduce the effects of fluctuating vehicle counts even more.

Two different reinforcement schemes were tested. For the fixed-distance and variable-distance rep-

resentations, using varying reinforcements helped to reduce the performance variation during learning and slightly improved performance, but not as much as was hoped. Varying reinforcements had almost no effect on the count representation and drastically reduced the performance of the count/duration representation.

For most of the tests performed where the SARSA representations performed better than the best fixed-duration strategy, the performance advantage was not that great. The constant intersection spacing of 440 feet may have helped the fixed-duration strategy. It has been stated (Oglesby 1975) that efficient synchronized traffic flow between two directions of traffic using fixed-duration timing is only possible if the spacing between intersections is approximately constant. Constant intersection spacing does not occur often in real life. If the intersection spacing had been varied in the simulation, the SARSA representations should have performed better than the fixed-duration strategy results that are reported here.

The methods studied here, while they worked very well in this simulation, would probably not work well if actually implemented. Real-world traffic controllers do not have complete knowledge of vehicle locations and do not terminate nicely as the simulator does. Reinforcement learning techniques applied in other areas could be tested. Auto traffic is somewhat similar to network traffic and data packets flowing through networks and reinforcement learning techniques for network data flow could be applied to traffic light controllers to see how they perform. The main difference is that data packets can continue to flow at varying rates, but are not required to physically stop as automobiles must to avoid collisions. The traffic control problem is similar to elevator dispatching (Crites and Barto, 1996). The arrival of vehicles at an intersection and the direction of travel are stochastic in nature. The elevator controller could be modified for the traffic control problem and, if successful, would be an implementation that could be refined and implemented in real traffic.

Since traffic flows continuously throughout the day, discounted reinforcements should be implemented so the controller can continue to learn while directing traffic over much longer simulations. The stochastic nature of the traffic flow needs to be considered by removing the knowledge of vehicle locations from the learning agent. Input from sensors that are installed at intersections could be used to predict

lane densities and vehicle locations. A connectionist representation utilizing sensor information from neighboring intersections, speed limits and lengths of lanes connecting the intersection and incorporating delays may out-perform the hard-coded representations tested here. A neural network can learn how best to partition the state-action space for optimal performance. A linear programming problem could be set up to estimate the number of vehicles in lanes by reducing the error of several simultaneous equations that account for vehicle turns and realizing that the number of vehicles entering an intersection must also exit the intersection.

Acknowlegement

Special thanks to Larry Dempsey, Joyce Garcia, Rich Kelly, Ann Muret, Randy Caner and Lance Walker who volunteered cpu time on their workstations to run and rerun the simulations used for this report.

References

Anderson, Charles W. (1988) Strategy Learning with Multilayer Connectionist Representations, GTE Labs TR 87-509.3, Waltham, MA

Crites, Robert H., Barto Andrew G. (1996) Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8.* Cambridge, MA: MIT Press., pp. 1017-1023.

Garnaas, Steve, "120th is giving commuters the green light", Denver Post, May 10, 1996, p. 1B and "Caught in Traffic, Timing the thing for Colorado's signal supervisor", p. 2B.

Ho, Fu-Sheng & Ioannou, Petros (1996) Traffic flow modeling and control using artificial neural networks. *IEEE Control Systems*, Oct 1996, 16(5):16-26.

Oglesby, Clarkson H., (1975) Highway Engineering, 3rd Edition, New York, John Wiley and Sons, pp. 332-338.

Olsson, G. (1996) Fewer traffic jams thanks to computers. At http://www.stockholm.se/bm/projects/-vagtrafiken.html.

Rumelhart, D. E., Hinton, G. E., & Williams (1986), Learning internal representations by error back propogation. In Rumelhart, D. E., Hinton, & R.Mclelland J. L. (1986), Parallel Distributed Processing: explorations in the microstructure of cognition, Cambridge, MA: Bradford Books, pp. 318-362.

Singh, S.P. and Sutton, R.S. (1996) Reinforcement learning with replacing eligibility traces. *Machine Learning* 22(1/2/3):123-158.

Sutton, Richard S. (1988) Learning to Predict by the Methods of Temporal Differences, *Machine Learning* 3(1):9-44, Boston, MA: Kluwer Academic Publishers

Sutton, R.S. (1996) Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, Cambridge, MA: MIT Press., pp. 1038-1044.

Tesauro, Gerald (1995) Temporal Difference Learning and TD-Gammon, *Communications of the ACM*, March 1995, 38(3):58-68.

Thorpe, T. (1997) A physically-realistic simulation of vehicle traffic flow. Technical Report 97-104,

Department of Computer Science, Colorado State University.