# Simulation and Optimization of Traffic in a City

Marco Wiering, Jilles Vreeken, Jelle van Veenen, and Arne Koopman

***Abstract*— Optimal traffic light control is a multi-agent decision problem, for which we propose to use reinforcement learning algorithms. Our algorithm learns the expected waiting times of cars for red and green lights at each intersection, and sets the traffic lights to green for the configuration maximizing individual car gains. For testing our adaptive traffic light controllers, we developed the Green Light District simulator. The experimental results show that the adaptive algorithms can strongly reduce average waiting times of cars compared to three hand-designed controllers.**

## I. INTRODUCTION

This paper describes our research in simulation and optimization of traffic light controllers to minimize car waiting times in a city. For optimization, we have developed a reinforcement learning (RL) [1], [2] algorithm that learns waiting times of cars before traffic lights, and uses these estimates to set the traffic light configuration. Our controller uses information about the locations of cars around the intersection, which is different from fixed-cycle traffic controllers where a cycle of configurations is defined in which all traffic gets a green light at some point.

We have compared our reinforcement learning algorithm to other hand-designed controllers using the Green Light District (GLD) simulator which we have developed. GLD makes it possible to construct infrastructures by using the mouse, simulate many different traffic patterns, build and test many different traffic light controllers, and display experimental results using a number of different statistical measures (such as average waiting time). The results showed that the reinforcement learning algorithms clearly outperform the handwritten controllers.

**Outline.** In section 2 we discuss reinforcement learning as an optimization tool, and present our traffic light controller based on reinforcement learning. In section 3 we describe the GLD simulator. Then, in section 4 we present our experimental setup and results. Section 5 summarizes related work on intelligent traffic light control. Finally, section 6 concludes this paper.

## II. REINFORCEMENT LEARNING FOR TRAFFIC LIGHT CONTROL

Reinforcement learning is a machine learning algorithm used for learning to control an agent from trial and error [1], [2]. The agent receives inputs from the environment describing the environmental state and uses its policy to select actions based on its current inputs. Evaluative feedback for judging how well the agent is doing is provided by a reward function which emits a scalar reward value after each time step. Now the goal of the agent becomes to learn a policy which maximizes its cumulative reward intake throughout its future. To learn the policy, a value function is used which estimates how much future reward the current state will give. Reinforcement learning has already been successfully applied for particular problems such as learning to play backgammon at human expert level [3], learning to control multiple elevators in a simulated building [4], and network routing [5].

**Decision Problem.** The environment consists of states $s \in \{S_1, \ldots, S_n\}$, for example for each discrete possible position (cell) in an infrastructure there is a state which may be occupied by a car. If there would be $m (m << n)$ cars which could occupy any state, there would be about $n^m$ possible traffic configurations.

There are actions for the traffic light controller that represent possible traffic light configurations that do not lead to any possible collisions between cars on an intersection. As an example: for a 4 four-way junction with one lane for going left and one lane for going right/straight, we have 8 possible actions (4 actions: for one street with two lanes both traffic lights green for left and for right/straight; 2 actions: from opposing streets both lights green for left; 2 actions: from opposing streets both lights green for straight/right). We denote $A_j$ as the traffic light configuration (action) for node $j$ in the infrastructure. The complete set of actions of all traffic controllers in the infrastructure is represented as the Cartesian product of the actions of all traffic nodes.

For developing the goal for the multi-agent system, we use a reward function. The reward function we use is more a cost function, since we deal with a minimization problem — our goal is to minimize the waiting times of all cars before traffic lights. The simple reward function, which we use, emits a cost of 1 if a car has to wait: $R(s, s) = 1$, and if a car can drive on the reward function emits a 0. Now the goal is to minimize the sum of the emitted costs during an extensive period of time.

For optimization, we use value functions. We define the value $Q(s, l)$ of a car being in one state (which may include the destination address, which makes the state $s = [p, d]$ with $p$ the position in the infrastructure, and $d$ as the car's destination) in the queue and having its light $l$ set to red/green as follows:

$$Q(s, red) = R(s, s) + \gamma V(s) = 1 + \gamma V(s)$$

and

$$Q(s, green) = \sum_{s'} P(s, s')(R(s, s') + \gamma V(s'))$$

All authors are with the Intelligent Systems Group, Institute of Information and Computing Sciences, Utrecht University, Padualaan 14, 3508 TB Utrecht, The Netherlands. For correspondence email marco@cs.uu.nl

Where $P(s, s')$ is the transition probability that the car goes to a next state $s'$. Note that $s'$ could also be $s$, since a car may have to wait even though its traffic light is green (as in congestions). We use $\gamma < 1$ for the discounting factor which trades off immediate costs versus long-term costs and makes the Q-functions finite. Finally, $V(s)$ is defined as:

$$V(s) = P(green|s)Q(s, green) + P(red|s)Q(s, red)$$

where $P(l|s)$ with $l \in \{red, green\}$ is defined as the probability the light is on red/green when the car is in state $s$. $V(s)$ estimates the average time a car has to wait until its destination address without knowing the traffic light decision. By definition $V(s) = 0$, if $s$ is the car's destination address (and the car will be removed from the infrastructure).

For all cars the waiting time for a red light will be longer than the waiting time for a green light. Therefore each car will have an advantage to set its light to green based on its waiting time for red minus its waiting time for green. If we take for each traffic node the sum of all gains of cars standing in the queue for the traffic light, we get an optimal local decision if we choose the traffic configuration $A$ that sets the light to green for traffic lights $i$ maximizing the cumulative gain:

$$A_j^{opt} = \max_{A_j} \sum_{i \in A_j} \sum_{s \in queue_i} Q(s, red) - Q(s, green)$$

By learning the waiting times, we get a better and better traffic light controller. How can we learn the transition probability functions $P(s, s')$, $P(green|s)$, and $P(red|s)$? This is simply done by frequency counting. Each time a car makes a transition to another state or has to wait, the probabilities are updated using counters, which is a simple and cheap way of inducing the model.

By applying this, we can compute both the Q- and V-functions (we initialize them to 0.0 for all states). Each time a car makes a step (or stands still), we update all counters and recompute the Q- and V-functions. If a traffic light decision has to be made, we sum over all gains of cars standing in the queue and select the configuration that maximizes the score.

**Co-learning driving policies.** A nice additional feature of our method is that cars can use the waiting times computed at traffic lights for selecting a way for driving in the city. For this we have to expand the state of the car by combining it with its destination. When we use a random shortest path driving policy, we select a random next lane from the set of possible next lanes lying approximately on the shortest path. In the co-learning approach cars use the V-function to choose the next lane. For this, we look what the average waiting time until the destination address for all possible next lanes are (in our system we use the first position of each next lane lying on a shortest path).

**Workings of the learning controllers.** We can compare our system to a system which always counts the number of cars which can move on when the light is set to green and sets the traffic light configuration which maximizes this count. In our system the counts are the learned Q-values which have values depending on the transition probabilities, and the probability of setting a light to green. For example, if one car competes with always 10 cars on a different lane, then the counting system will always let the 10 cars drive and the single one wait. In the RL system, the gain of the single car will go to a very large value if it has to wait a long time. Therefore the single car is allowed to drive after some time as well, which is an effect of the learning dynamics and not designed ad-hoc.

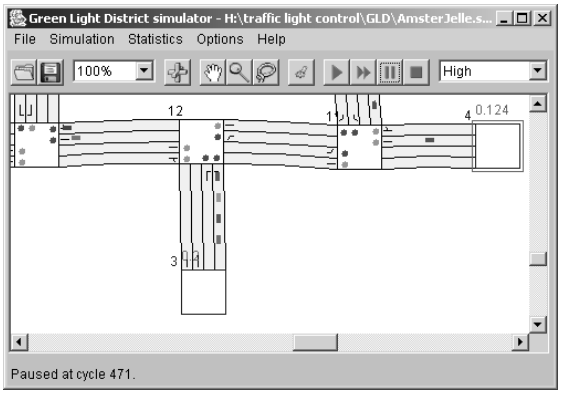


Fig. 1. A screenshot from the Green Light District simulator. The image shows a junction with traffic lights, two edge-nodes, and some roads with cars. The number above the edge-node on the right indicates its spawning frequency.

## III. GREEN LIGHT DISTRICT SIMULATOR

For simulating our adaptive controller and comparing it with other algorithms on a broad range of infrastructures and traffic patterns, we developed the Green Light District (GLD) simulator in Java. [1] GLD allows us to edit infrastructures using the mouse, to set different spawning frequencies creating different traffic patterns, to compare a large number of implemented controllers (new ones can be easily added), and to evaluate the controllers using different statistical measures (such as average waiting time of cars). The simulator itself is based on a cellular automaton model [6], and therefore a microscopic model which can be used for modelling various amounts of detail (e.g. the road user could be a car, bus, police car, etc.).

**Infrastructures.** An infrastructure consists of roads and nodes. A road connects two nodes, and can have several lanes in each direction (see Fig. 1). The length of each road is expressed in units. A node is either a junction where traffic lights are operational (although when it connects only two roads, no traffic lights are used), or an edge-node.

**Agents.** There are two types of agents that occupy an infrastructure; vehicles and traffic lights. All agents act autonomously, following some simple rules, and get updated every time-step. Vehicles enter the network at the

[1] GLD can be downloaded from http://www.sf.net/projects/stoplicht.

edge-nodes. Each edge-node has a certain probability of generating a vehicle at each time step. Each vehicle that is generated is assigned a destination, which is one of the other edge-nodes. The distribution of destinations for each edge-node can be adjusted. There are several types of vehicles, defined by their speed, length, and number of passengers. For our experiments we only used cars, which move at a speed of two units (or one or zero if they have to brake) per time step, have a length of two units, and have two passengers. The state of each vehicle is updated every time step. It either moves with the distance given by its speed, or stops when there is another vehicle or a red traffic light ahead. At a junction, a car decides to which lane it should go next according to its driving policy. Once a car has entered a lane, it cannot switch lanes.

**Controllers.** Every junction is controlled by a traffic light controller (TLC) that decides on the best configuration of red and green lights. A TLC will only consider safe configurations, that is, configurations in which moving cars do not intersect. A TLC can share information with other controllers to improve global performance. GLD has several built in TLCs, and allows for custom TLCs. We used the following controllers in our experiments:

- The **TC-1 algorithm** uses the reinforcement learning algorithm described in section 2. The car-state is extended with information about the destination of the car. The discount factor used by the algorithm was set to 0.9.
- **TC-1 Destinationless** is a simplified version of TC-1 that does not use destinations in its computations.
- **TC-1 Bucket** is TC-1 extended with the Bucket algorithm that is described below.
- **TC-1 Co-learning** can be used with TC-1 and with TC-1 Bucket, but not with TC-1 Destinationless, since it requires information about vehicle destinations.
- **Best first** always selects the traffic light configuration which sets the lights to green for the largest amount of cars in the queues.
- **The Bucket algorithm** is used for optimizing traffic flow in cities of very high traffic densities. The basic thought underlying the mechanism is that each traffic light from which no movement is possible due to overfull destination lanes communicates part of its summed gain values there to stimulate movement. Each traffic light sums the gain values calculated by the traffic controller algorithm into its bucket. This value is used to determine the optimal configuration of green traffic lights, instead of the originals calculated by the TLC.
- **ACGJ-3** was the first traffic light controller algorithm we paired with the bucket mechanism. The gain value for a traffic light is calculated by summing over all waiting road users, multiplying the road user weight with a length-factor $f^n$, where $n$ is the place of the car in the queue. In the experiments presented in this paper we used the ACGJ-3 algorithm (with $f = 1$), resembling the Best First algorithm paired with the bucket mechanism.
- **Relative Longest Q** chooses to set the lights to green for the queues with the maximum ratio of waiting road users versus length of the road. Because completely filled up roads get priority over roads where many people are waiting, but are not full, this algorithm may avoid jammed traffic.

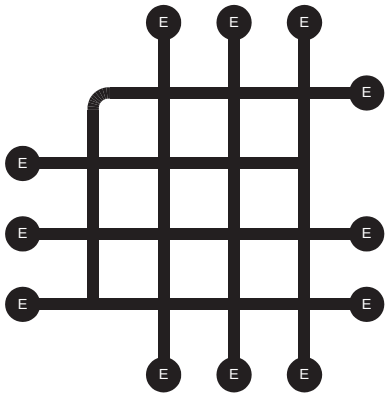


Fig. 2. Infrastructure for the experiment. Nodes marked with E represent edge-nodes. All intersections are controlled by traffic lights, and all edges represent roads with two lanes in each direction. At each edge node, 0.4 cars are generated each cycle.

As driving policies, the random shortest path policy selects a random next lane lying on an approximately (within $10\%$ range) shortest path to the destination address. We compare this to co-learning which uses the computed waiting times, which can only be used with the RL algorithm.

## IV. EXPERIMENTS

We performed two series of experiments. For our first series of experiments we used the grid-like infrastructure depicted in Fig. 2. With 12 edge nodes and 16 junctions, 15 of which have traffic signs, and 36 roads, 4 lanes wide each, it is a structure of reasonable size and interesting enough to compare the different algorithms. When totally filled up, it contains about 1500 vehicles. All edge nodes were set at the same spawning frequencies of 0.4, and for each edge node, all other nodes were given an equal chance of being the destination of a new car.

**Results.** As shown in Fig. 3, the TC-1 algorithms outperform all other algorithms. TC-1 with co-learning clearly performs best, with an average trip waiting time (ATWT) of 6.46 during the last 10.000 cycles (see Table I). TC-1 Destinationless, the simplified version of TC-1, performs surprisingly well, and seems to outperform TC-1 when co-learning is not used. The TC-1 bucket algorithm is the worst performing of the TC-1 algorithms, but still outperforms ACGJ-3, which is a best first algorithm with a bucket. Note that the best non-adaptive controller (ACGJ-3) has an average trip waiting time which is more than 30% larger than the ATWT of the best TC-1 algorithm.

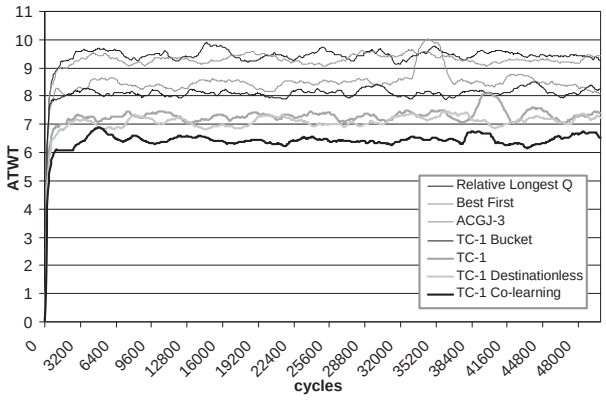


Fig. 3. Results of the experiment. Each result is an average of ten simulations.

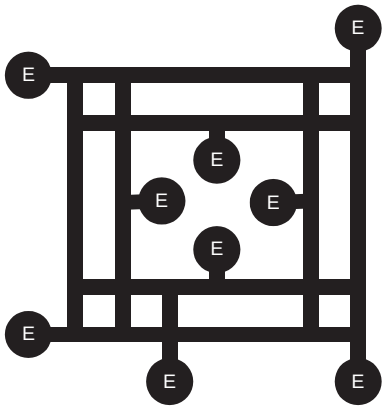


Fig. 4. Infrastructure for the second experiment. The inner edge-nodes represent a city centre, surrounded by ring roads. Traffic from outside the city can either drive around or enter the city centre. Traffic coming from the centre always leaves the city. All intersections feature traffic lights.

**Second Experiment.** We also compared the traffic light controllers using an infrastructure containing an inner and an outer part of edge-nodes, resembling a city-centre surrounded by highways (see Fig. 4). The inner edge-nodes represent access to the city centre, and the outer edge-nodes are the connections to the approach roads. The network is of reasonable size, with a total of 615 units length of two-lane, two-way roads. We again determined a traffic load near saturation by experimenting with different spawning frequencies. For this infrastructure, this point occurred when all spawning frequencies are set to $0.4$.

**Results of Second Experiment.** The results of the second experiment are shown in Table II and Fig. 5. The TC-1 algorithms again outperformed the others, but TC-1 with co-learning is no longer the best. TC-1 with the normal shortest path driving policy performs better, indicating that the effect of co-learning depends on the infrastructure. In this particular structure, where there are few origins and destinations, but always multiple routes, choosing between routes randomly may result in an equal distribution of vehicles. When co-learning is used, all drivers with the same origin and destination choose the same route, and that route might get more saturated. Another interesting result is the fact that the TC-1 destinationless algorithm, which is a simplified version of TC-1, performs as well as the TC-1 algorithm. Note that the best non adaptive controller (again ACGJ-3) has an ATWT which is more than 25% larger than the ATWT of the best RL algorithm.

TABLE I

COMPARISON OF THE CONTROLLERS ON THE INFRASTRUCTURE OF FIG. 2. THE ATWT SHOWN IS AN AVERAGE OVER THE LAST 10.000 CYCLES OF ALL TEN EXPERIMENTS

| | Controller | ATWT |
|---|---|---|
| 1 | TC-1 Co-learning | 6.46 |
| 2 | TC-1 Destinationless | 7.14 |
| 3 | TC-1 | 7.40 |
| 4 | TC-1 Bucket | 8.20 |
| 5 | ACGJ-3 | 8.46 |
| 6 | Best First | 9.27 |
| 7 | Rel Longest Q | 9.46 |

TABLE II

RESULTS OF THE EXPERIMENT ON THE CITY-LIKE INFRASTRUCTURE DEPICTED IN FIG. 4.

| | controller | ATWT |
|---|---|---|
| 1 | TC-1 Destinationless | 2.67 |
| 2 | TC-1 | 2.68 |
| 3 | TC-1 Co-learning | 2.82 |
| 4 | TC-1 Bucket | 3.25 |
| 5 | ACGJ-3 | 3.57 |
| 6 | Best First | 4.23 |
| 7 | Rel Longest Q | 4.75 |

## V. RELATED WORK

Many different approaches to intelligent traffic light control exist. In this section we summarize a number of these.

**Expert Systems.** An expert system uses a set of given rules to decide upon the next action. In traffic light control,

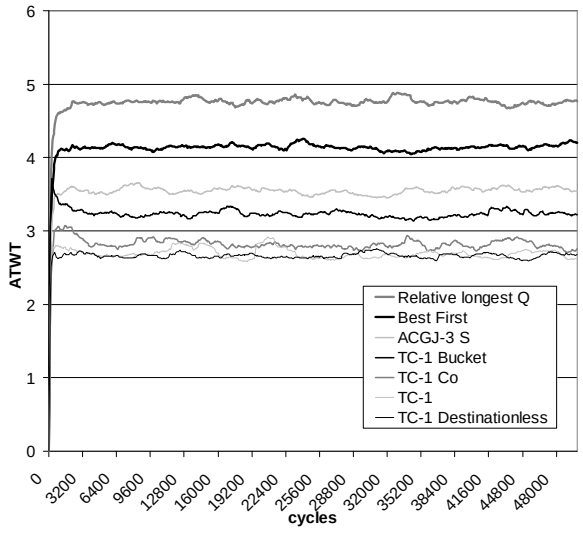


Fig. 5. Results of the experiment on the city-like infrastructure depicted in Fig. 4. Each result is an average of 10 simulations.

such an action can change some of the control parameters. Findler and Stapp (1992) describe a network of roads connected by traffic light-based expert systems. The expert systems can communicate to allow for synchronization. Performance on the network depends on the rules that are used. For each traffic light controller, the set of rules can be optimized by analyzing how often each rule fires, and the success it has. The system could even learn new rules. Findler and Stapp showed that their system could improve performance, but they had to make some simplifying assumptions to avoid too much computation.

**Prediction-based optimization.** Tavladakis and Voulgaris (1999) describe a traffic light controller using a simple predictor. Measurements taken during the current cycle are used to test several possible settings for the next cycle, and the setting resulting in the least amount of queued vehicles is executed. The system seems highly adaptive, and maybe even too much so. Since it only uses data of one cycle, it could not handle strong fluctuations in traffic flow well. In this case, the system would adapt too quickly, resulting in poor performance.

Liu et al. (2002) introduce a way to overcome problems with fluctuations. Traffic detectors at both sides of a junction and vehicle identification are used to measure delay of vehicles at a junction. This is projected to an estimated average delay time using a filter function to smooth out random fluctuations. The control system tries to minimize not only the total delay, but the summed deviations from the average delay as well. Since it is no longer beneficial to let a vehicle wait for a long time, even if letting it pass would increase the total waiting time, this introduces a kind of fairness. Data of about 15 minutes is used to determine the optimal settings for the next cycle, and even using a simple optimization algorithm, the system performs well compared to preset and actuated controllers.

**Fuzzy Logic.** Tan et al. (1995) describe a fuzzy logic controller for a single junction that should mimic human intelligence. Fuzzy logic offers a formal way of handling terms like ”more”, ”less”, ”longer” etc., so rules like ”if there is more traffic from north to south, the lights should stay green longer” can be reasoned with. The fuzzy logic controller determines the time that the traffic light should stay in a certain state, before switching to the next state. The order of states is predetermined, but the controller can skip a state if there is no traffic in a certain direction. The amount of arriving and waiting vehicles are quantized into fuzzy variables, like many, medium and none. The activation of the variables in a certain situation is given by a membership function, so when there are 5 cars in the queue, this might result in an activation of 25% of ’many’ and 75% of ’medium’. Fuzzy rules are used to determine if the duration of the current state should be extended. In experiments the fuzzy logic controller showed to be more flexible than fixed controllers and vehicle actuated controllers, allowing traffic to flow more smoothly, and reducing waiting time. A disadvantage of the controller seems to be its dependence on the preset quantification values for the fuzzy variables. They might cause the system to fail if the total amount of traffic varies. Furthermore, the system was only tested on a single junction.

Lee et al. (1995) studied the use of fuzzy logic in controlling multiple junctions. Controllers received extra information about vehicles at the previous and next junctions, and were able to promote green waves. The system outperformed a fixed controller, and was at its best in either light or heavy traffic. The controller could easily handle changes in traffic flow, but required different parameter settings for each junction.

**Evolutionary Algorithms.** Taale et al. (1998) compare using evolutionary algorithms (a $(\mu, \lambda)$ evolution strategy [12]) to evolve a traffic light controller for a single simulated intersection to using the common traffic light controller in the Netherlands (the RWS C-controller). They found comparable results for both systems. Unfortunately they

did not try their system on multiple coupled intersections, since dynamics of such networks of traffic nodes are much more complex and learning or creating controllers for them could show additional interesting behaviors and research questions.

**Reinforcement Learning.** Reinforcement learning for traffic light control has first been studied by Thorpe [14], [15], but Thorpe's approach is different from ours. He used a traffic light-based value function, and we used a car-based one. Thorpe used a neural network for the traffic-light based value function which predicts the waiting time for all cars standing at the junction. This means that Thorpe's traffic light controller have to deal with a huge number of states, where learning time and variance may be quite large. Furthermore, Thorpe used a somewhat other form of RL, SARSA (State-Action, Reward-State Action) with eligibility traces [16], and we use model-based RL.

Thorpe trained only a single traffic light controller, and tested it by instantiating it on a grid of $4 \times 4$ traffic lights. The controller can decide to let either traffic on the north-south axis or on the east-west axis pass. A neural network is used to predict the Q-values for each decision, based on the number of waiting cars and the time since the lights last changed. The goal state is the state in which there are no cars waiting.

The system outperformed both fixed and rule-based controllers in a realistic simulation with varying speed. Performance is said to be near optimal. Controllers trained on a single junction had the same performance as controllers trained in a network.

## VI. CONCLUSION

We presented our adaptive reinforcement learning algorithms for learning to control traffic lights. These algorithms make their decisions based on the amount of traffic around an intersection and are therefore different from commonly used fixed-cycle controllers. The RL algorithms adapt the traffic light controllers automatically without the need for manual tuning. Furthermore they are able to deal with non-stationary traffic patterns. For comparing the RL algorithms to other handwritten controllers, we developed the GLD simulator which makes it easy to compare the algorithms on different infrastructures and traffic patterns. The simulations show that the RL algorithms reduce average waiting times with more than $25\%$ on crowded traffic compared to manually designed non-adaptive controllers.

In future work, we want to study the use of green waves in the RL algorithms. We want to do this by using communication between traffic lights. Furthermore, we want to study other controllers and compare them to the ones presented here. We also want to study different traffic patterns, including non-stationary ones. Finally we want to make the traffic model more realistic by taking different car-speeds into account. The RL algorithms can use this information immediately by using the speeds in the states of the cars.

## REFERENCES


[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT press, Cambridge MA, A Bradford Book, 1998.

[2] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[3] G.J. Tesauro, "Temporal difference learning and TD-Gammon," *Communications of the ACM*, vol. 38, pp. 58–68, 1995.

[4] Robert H. Crites and Andrew G. Barto, "Improving elevator performance using reinforcement learning," in *Advances in Neural Information Processing Systems*, David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, Eds. 1996, vol. 8, pp. 1017–1023, The MIT Press.

[5] Justin A. Boyan and Michael L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in Neural Information Processing Systems*, Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, Eds. 1994, vol. 6, pp. 671–678, Morgan Kaufmann Publishers.

[6] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *J. Phys. I France*, vol. 2, pp. 2221–2229, 1992.

[7] N. Findler and J. Stapp, "A distributed approach to optimized control of street traffic signals," *Journal of Transportation Engineering*, vol. 118-1, pp. 99–110, 1992.

[8] K. Tavladakis and N. C. Voulgaris, "Development of an autonomous adaptive traffic control system," in *ESIT '99 - The European Symposium on Intelligent Techniques*, 1999.

[9] H. L. Liu, Jun-Seok Oh, and W. Recker, "Adaptive signal control system with on-line performance measure," in *81st Annual Meeting of the Transportation Research Board*, 2002.

[10] K. K. Tan, M. Khalid, and R. Yusof, "Intelligent traffic lights control by fuzzy logic," *Malaysian Journal of Computer Science*, vol. 9-2, 1995.

[11] J.H. Lee, K.M. Lee, K.A. Seong, C.B. Kim, and H. Lee-Kwang, "Traffic control of intersection group based on fuzzy logic," in *Proceedings of the 6th International Fuzzy Systems Association World Congress*, 1995, pp. 465–468.

[12] I. Rechenberg, "Evolution strategy: Nature's way of optimization," in *Methods and Applications, Possibilities and Limitations*, Bergmann, Ed., 1989, pp. 106–126, Lecture notes in Engineering.

[13] H. Taale, Th. Bäck, M. Preuß, A. E. Eiben, J. M. de Graaf, and C. A. Schippers, "Optimizing traffic light controllers by means of evolutionary algorithms," in *EUFIT'98*, 1998.

[14] T. L. Thorpe and C. Andersson, "Traffic light control using sarsa with three state representations," Tech. Rep., IBM corporation, 1996.

[15] Thomas Thorpe, "Vehicle traffic light control using sarsa," M.S. thesis, Department of Computer Science, Colorado State University, 1997.

[16] Richard S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," *Advances in Neural Information Processing Systems*, vol. 8, 1996.