Alireza Fasih, Umair Ali Khan, Jean Chedjou, Kyandoghere Kyamakya

Transportation Informatics Group, Alpen Adria University, Klagenfurt, Austria.

Traffic Light Controller by Reinforcement Learning Method with Local States

***Abstract** – In this paper we describe an efficient method for traffic light controllers. This method is based on enhanced reinforcement learning with local states around each traffic light on crossings. It uses many independent controllers depending upon the number of crossings. Each controller can learn the best actions by a supervised method that is based on reward and punishment policy. The supervisor system monitors the outcome of actions on the specific state. The reward policy is based on this monitoring and tries to minimize the traffic on crossings. Each node or agent tries to self-organize for getting the best decisions (actions) for each state. It is possible to implement this project by the minimum infrastructure and accessories. At the end some result and benchmarking with classical method has shown.*

Key words Traffic simulator, reinforcement learning, multi agent systems.

Introduction

Traffic control in urban areas that have a very complicated dynamic vehicles flow is one of the important issues. Transportation research has the target to optimize transportation flow of vehicles. As the number of roads and infrastructures are limited, the intelligent traffic controllers and crossing junctions will become a very important issue for the future. In the same context, we have designed an intelligent controller based on reinforcement learning for controlling crossing junctions like a multi agent system, separately.

Reinforcement Learning

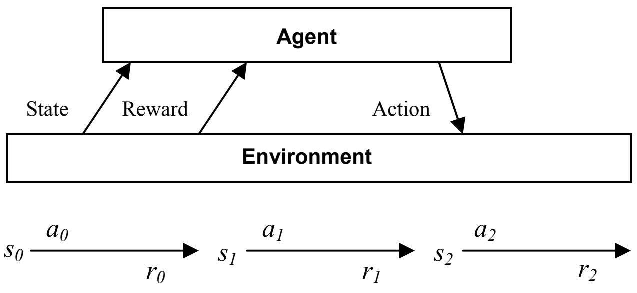
Machine learning algorithm falls into three main categories. *Supervised Learning* in which an *agent* (learner) gets a goal or target from the environment for every input which specifies what relevant response the agent should generate for this input. The agent then adjusts its actual response according to the specified response so that it is more likely to produce an output closer to or equal to the specified response the next time it receives the same input.
Unsupervised Learning in which an agent can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. By contrast with supervised learning, there are no explicit target outputs or environmental evaluations associated with each input [1].
The third category is *Reinforcement Learning* which is closer to supervised learning in the sense that the agent receives feedback about the appropriateness of its output. However, as far as the errors are concerned, the two approaches differ remarkably. On the basis of agent's response, the supervised learning needs a teacher which tells the agent what it should have done. Whereas, the reinforcement learning only tells how appropriate or inappropriate the response is by assigning a certain scalar value to the response. The scalar value is higher (or positive) for an appropriate response; and lower (or negative) for an in appropriate response. In this sense, reinforcement learning is more independent and mimics natural learning by trial-and-error.

Reinforcement learning dates back to the early days of cybernetics and work in statistics, psychology, neuroscience, and computer science. In the last five to ten years, it has attracted rapidly increasing interest in machine learning and artificial intelligence communities. Reinforcement learning is a way of programming agents by reward and punishment without needing to specify how the task is to be achieved [2].

In reinforcement learning, each agent has a set S of states and a set A of appropriate actions. For each state S_i, the agent can choose any of the appropriate actions A_i based on a control policy. The response generated by the agent by taking an action can then be rewarded or punished by a reward-punish function. The most fundamental elements of reinforcement learning systems are *a policy, a reward-punish function* and *a value function*.

A policy determines the mapping of a state to an appropriate action. The policy varies from being simple to very computation expensive and remains stochastic throughout the learning process. An agent keeps changing its mapping policy based on the reward/punishment.

A *reward-punish function* determines the quality of agent's response to a given input in terms of its being appropriate or inappropriate and assigns a scalar value to the response. The scalar value, being very low indicates the inappropriateness of the output and is called *punishment*. Whereas a higher scalar value reflects a good response and is called a *reward*. Like policy, the punish-reward function may be stochastic.



Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + ..., \text{ Where } 0 \le \gamma < 1$$

Fig-1 An agent interacting with its environment

The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states [3]. Value is the most important element of reinforcement learning which needs to be appropriately judged, as all the actions are taken according to t he values judgment. This is the element which the agent is interested to maximize in order to achieve a specific target.

An agent interacting with an environment with a possible set of states S and a possible set of actions A is shown in Fig- 1 [4]. As shown in Fig-1, in each stae S_i, the agent performs an action a_i and gets a reward r_i. The aim is to learn a control policy that maximizes the expected sum of these rewards.

Reinforcement Learning based Traffic light controller

One of the main problem areas in traffic light controllers is non-stationary vehicle flow patterns. Our designed controller is based on some assumptions that are in accordance with the real world situations. The assumptions are:

1- Cars move on a specific path during the simulation phase. On the other hand the start and end position is predefine in simulation.
2- All states are local for controller (RL1-TLC). Also we quantized the state of each road that leads to a crossing to 4 levels.
3- For each traffic light, time interval is constant. This reduces the number of actions to 6 for each traffic light as shown in Fig-1.

One of the most important parts of our project is traffic simulator. We developed a microscopic simulator for training controller and analyzing situations. The simulator allows us to change/tune some essential parameters; like vehicles flow rate, control signals, acceleration and deceleration of vehicles. These parameters need to be correctly identified as they can add noise in the system. Also they enable users to model the microscopic behavior of drivers in a large area[5-7].

The simulator produces 2 feedbacks for controller and it can receive actions from controller for applying on the model. The first feedback is number of cars behind the red light on the queue, and another feedback is waiting time for each car that is stopped in the queue.

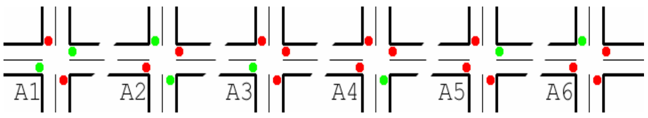


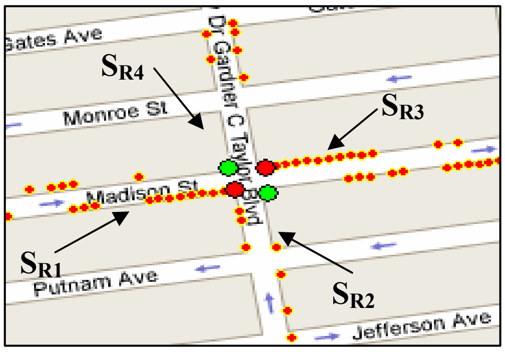
Fig-2 List of Actions for each Cross junction

As the traffic light controller deals with highly dynamic and non-stationary flow patterns of traffic, it is very hard to simulate. In this context, we designed a distributed controller composed of various independent controllers which deal with separate and individual working of each traffic light like a multi-agent system. The main benefit of this architecture is that when one of the traffic lights fails or the system is saturated, other controllers lead the system to a balanced situation.

After learning process, each controller monitors the situation of the crossing, and selects the best action (action with the best reward on that situation) according to the action list. By this action list controller can select the best action for that situation to handles the traffic.

The Reinforcement learning method is based on reward-and-punish policy. Each agent or light controller on crossing can improve itself by getting reward and punish values. The main improvement in this controller is the enhancement of reward policy method. Because of the complex dynamics in traffic model, we are not able to estimate the next state of crossing after doing an action. Therefore, the monitor evaluates the situation of crossing depending on the previous state. If the environment is better than the previous state, the system rewards a positive value. The value has a real domain and it rewards/punishes depending on the situation. If the situation becomes better (less cars on the queue for all roads that leads to the crossing), the system will get a plus value as a reward and if the situation becomes worse the system will get a negative value as punishment. Controller has access to the number of cars that are behind the red light on the queue and also waiting time for cars [8-10].

Fig-3 shows the data structure of crossing as a S_{Ri} variable. State is a combination of these 4 strings. According to the table-1, each string can be low, medium, high or very high.



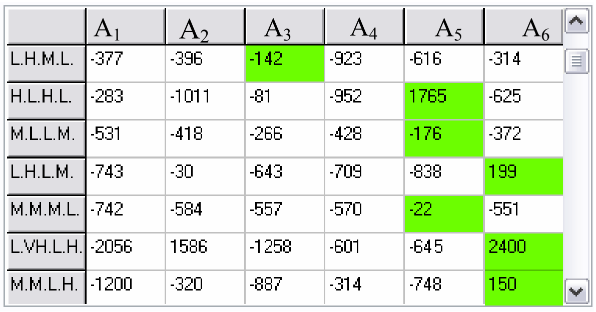
Crossing State = {S_{R1}, S_{R2}, S_{R3}, S_{R4}}

Fig-3 Cross State

The main problem of controllers based on reinforcement learning is initial states. In the phase of training, controller doesn't know which action is better. Therefore, at the beginning, system cannot predict the next situation. Hence, actions are selected randomly. The supervisor program puts the reward and punishment values in a table as shown in Fig-4. This table has 6 columns and many rows depend to the number of actions and comination of crossing state. Each column is created for saving a specific action value.

In this case there are 6 possibilities for selecting actions for any state. Controller should find the highest value in any state for selecting the action. If still there are some null value in columns it should be select randomly, otherwise the best action according to the highest value will select.

Another issue is number of states that are depending to the number of cars in the queue and waiting time. We have to discritize it to few limited domain.



	A_1	A_2	A_3	A_4	A_5	A_6
L.H.M.L.	-377	-396	-142	-923	-616	-314
H.L.H.L.	-283	-1011	-81	-952	1765	-625
M.L.L.M.	-531	-418	-266	-428	-176	-372
L.H.L.M.	-743	-30	-643	-709	-838	199
M.M.M.L.	-742	-584	-557	-570	-22	-551
L.VH.L.H.	-2056	1586	-1258	-601	-645	2400
M.M.L.H.	-1200	-320	-887	-314	-748	150

Fig-4 Some Rows from State-Action Table

For reducing the number of states, we must quantize the state values. In this case, the rate of learning will improve and the system will converge to a good situation rapidly.

In real world, quantizing of situations to few states it creates problem. Because at the beginning of the learning phase, the controller might fail and drivers will have to wait behind the red light without any reason. The rate of learning and size of state has a tradeoff; it means that by decreasing the number of states we can speed up the learning rate.

If we simulate the traffic model exactly same as real situation or close to it, we can train the controller in computer. After that, for tuning the controller we can switch to real world. We define the road states that lead to a crossing, as in Table-1. Depending on the amount of traffic in each road, we assign a Letter to that state. By this model, we have 4^4 States for each cross.

During the training phase in each state, controller can get a wrong decide at most n-1 time. That 'n' denotes the number of actions for cross junction.

Table-1, Road State

S_{Ri} = Road State	Number of vehicles on i^{th} Road
L=Low	< 5
M=Medium	5~15
H=High	15~40
VH=Very High	>40

For the getting a optimum solution we have to quantize the situation to many partially states. Therefore by increasing the number of 'n', training phase also will be increase.

Reward Policy

The policy of reward in this controller is not completely similar to the classical reinforcement learning method. In reinforcement learning, prediction of next step is needed and reward policy depends on the prediction result and next state result. Due to the lack of a determinant traffic model, we define our system to calculate current state reward after end of the next state. Amount of rewards depend on the waiting time of vehicles that are behind the red light and also number of cars, before and after applying the new action.

(1) $$W_{s,r} = \sum_{K=0}^{N_i} t_{s,r,K} \cdot N_{s,r}$$

In Eq-1, $W_{s,r}$ denotes summation of delay for all cars behind the red light for road number r in state s that leads to the crossing. $t_{s,r,k}$ is amount of counter for the car number *k* that is waiting on the queue of road number *r* and $N_{s,r}$ shows the number of cars on the queue for road number r on the crossing in the state of *s*. We have to calculate this formula at the end of each state. We denoted s as a number of states. In this model, each car has an internal counter. It counts the time delay for stopped car behind the red light. When the car starts to move, the counter will reset to zero.

(2) $$PR = \sum_{r=0}^{3} W_{s,r} - \sum_{r=0}^{3} W_{s-1,r}$$

Reward and punishment value will calculate by eq-2. In this equation, *PR* is denotes the reward and punishment result. In this form we have to subtract the result of current state to previous state. After normalizing the value, we assign it as a reward or punishment value for that specific action, in the state-action table.

Results

Fig-5 shows a snapshot from the simulator during the learning phase with a Google earth map in background. Controller is robust and reliable; it cans tune the controller for optimizing the traffic on crossing. A comparison of this method with the classical controller is shown in Fig-6. The figure shows that as compared to static timer controller and crowd-road-green (*First Crowded Lane*) controller, this method is much better and stable.

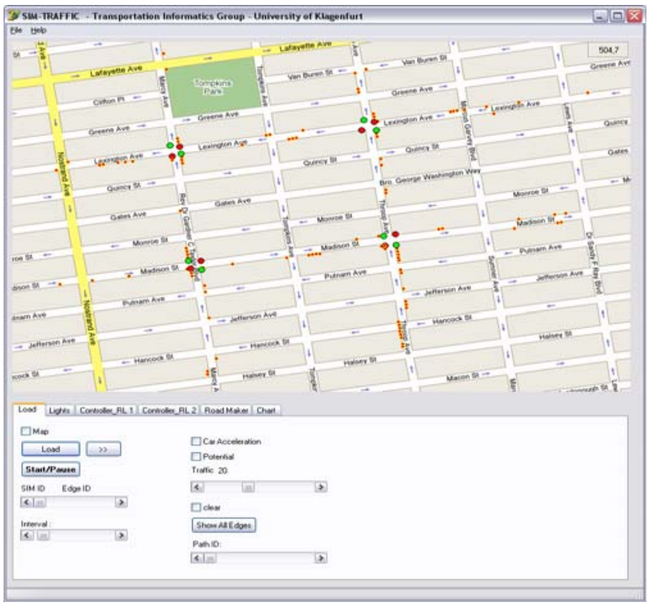


Fig-5 A snapshot from the simulator during learning phase

The comparison of RL Traffic Controller with *First Crowded Lane* and static timer controller shows that this method can manage the cars faster than static timer. And average of cars that stopped on the crossing is less than static timer. The average amount of stopped car behind the traffic light is 20~25.

A simple timer controller without any feedback that switches the lights can not manage the dynamic situation of crossing and by a simple shuck it fail and we have a jam on the crossing. *First Crowded Lane* has the worst result that always give a priority of passing to crowded road and blocks the others roads.

Single and independent RL controller is robust and it can control the traffic jam at the minimum time. It's ideal for microscopic traffic controller. For the macroscopic case we are going to couple it locally to other crossing. Because is not possible to controlling the traffic in macroscopic case by independent traffic controller [11-12].

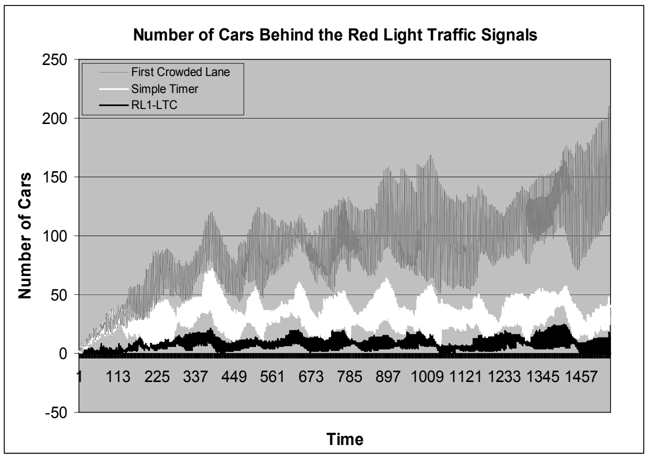


Fig-6 Comparison of Crowd-Road-green, Static Timer and RL Traffic Light Controller

Conclusion and future work

Our research shows that a distributed system based on RL is quite applicable as a compare to other classic method such as first crowded and static timer. We aim to design and implement some controller based on TD temporal difference, Q-learning and also Evolutionary Method. And also we are going to implement a FPGA based cellular automata traffic simulator for our research.

REFERENCES

[1] P. Dayan, "Unsupervised learning," *The MIT Encyclopedia of the Cognitive Sciences*, 1999
[2] L.P. Kaelbling, et al., "Reinforcement learning: A survey," *Arxiv preprint cs.AI/9605103*, 1996
[3] R.S. Sutton and A.G. Barto, "Reinforcement learning," *Journal of Cognitive Neuroscience*, vol. 11, no. 1, 1999, pp. 126-134.
[4] T.M. Mitchell, "Machine Learning. WCB," *Mac Graw Hill*, 1997, chapter 13, pp. 368.
[5] John A. Shanks, University of Otago, "Probability in Action: the Red Traffic Light" *Journal of Statistics Education* Volume 15, Number 1 (2007).
[6] Hoyer, R. and U. Jumar, *Fuzzy control of traffic lights.* Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence. Proceedings of the Third IEEE Conference on, 1994: p. 1526-1531.
[7] Wiering, M., et al., *Intelligent traffic light control.* ERCIM News, European Research Consortium for Informatics and Mathematics, 2003. 53: p. 40–41.
[8] Wiering, M., *Multi-Agent Reinforcement Leraning for Traffic Light Control.* Proceedings of the Seventeenth International Conference on Machine Learning table of contents, 2000: p. 1151-1158.
[9] Thorpe, T.L. and C. Andersson, *Traffic light control using sarsa with three state representations*.
[10] Bingham, E., *Reinforcement learning in neurofuzzy traffic signal control.* European Journal of Operational Research, 2001. 131(2): p. 232-241.
[11] Abdulhai, B., R. Pringle, and G.J. Karakoulas, *Reinforcement Learning for True Adaptive Traffic Signal Control.* Journal of Transportation Engineering, 2003. 129: p. 278.
[12] Sekiyama, K., et al., *Self-organizing control of urban traffic signal network.* Systems, Man, and Cybernetics, 2001 IEEE International Conference on, 2001. 4.

Authors

Alireza Fasih[1], Umair Ali Khan[2], Dr. Chamberlian Chejou[3], Prof. Dr. Kyandoghere Kyamakya[4] Research Assistant Transportation Informatics Group, Alpen Adria University, Klagenfurt, Austria, Emails: {alireza.fasih, jean.chedjou, kyandoghere.kyamakya @ uni-klu.ac.at}, ukhan@edu.uni-klu.ac.at