Traffic Control through Upper Confidence Tree

Ga Wu

Student Id : 932-677-101
Course code: CS533

Contents

1 Objective Introduction **2**

2 Domain Description **3**

3 Cell Transmission Model **4**

4 Monte Carlo Tree Search **7**

5 Experiments **10**

6 Conclusion **13**

1 Objective Introduction

The goal of this project is to apply dynamic traffic light control through Monte Carlo Planning technique. Traffic control, especially traffic light control, is a complex markov decision processes problem. However, since the traffic condition is changing over time, the states of this MDPs is infinite and continuous. Learning complete transition function or reward function is almost impossible. Therefore, The basic off-line learning methods are inapplicable in this environment. Monte Carlo Planning has great online view: it compute policy for current state but not all. Another reason we prefer to use Monte Carlo Planning is that the traffic field has a lots of simulation theories, which make this efficient simulator based learning algorithm applicable.

However, not all monte carlo planning methods are preferred in traffic environment. For example, sparse sampling. Sparse sampling has been proved to be a great method that allow to get near optimal policy. But, this algorithm needs large time period to make single learning update, which cause it has bad anytime behavior. Real-time traffic control requires to make decision in relatively short time and can return an action anytime. Thus, we have to consider more efficient algorithm. In this project, we are going to use Upper Confidence Tree to get near optimal Monte Carlo approximation. However, we also notice that the traffic state is continuous, which prevent this powerful method to work directly. And, that will be a great challenge of this project.

In this project, we are going to use Cell Transmission Model to model traffic flow. We choose this relatively simple model mainly because of the computational ability of personal computer. The Monte Carlo Tree Search methods require large number of times sampling to get relatively reliable approximation. Complex modeling methods will either running out the memory or extending running time, and either of those will cause the Monte Carlo Tree Search gets bad result. Also, for the same reason, in this project, we only consider single direction roads.

2 Domain Description

Traffic control, especially traffic light control, is complex markov decision processes problem. It involves continuous states and concurrent actions. The result of transition from one state and one action is stochastic and even involve infinite possible result states for large traffic network. But, we still can describe it as follow:

States: In this project, since we only consider simple traffic network with four intersections, state is a vector of 16 elements: 12 road elements and 4 traffic light status elements. x_1 to x_{12} represent the 12 roads in the experiment as shown in Figure 2(b). Each of them denotes a vector of densities, and length of the vector also represents the length of that road(number of cell). Traffic light of intersections is in boolean type, since there are only two phrases for the intersections.

$$\mathbf{x} = \begin{bmatrix} x_1 : \mathbb{R}^{n_1} \\ x_2 : \mathbb{R}^{n_2} \\ \vdots \\ x_{12} : \mathbb{R}^{n_{12}} \\ x_{13} : \mathbb{B} \\ x_{14} : \mathbb{B} \\ \vdots \\ x_{16} : \mathbb{B} \end{bmatrix} \tag{1}$$

Actions: We only consider to manage traffic light control rather than others. The traffic light for a intersection has binary phrase. For one phrase, a road has 'right to go' and the other will be blocked. So, there are only two actions for single intersection, either 'HOLD' or 'CHANGE'. Although the actions for single intersection is quite simple, we notice that, for multiple intersections, the number of actions grows exponentially. In this project, we consider four intersections. Therefore, the action number is 16 as shown in equation 2.

$$\mathbf{a} = \begin{bmatrix} a_1 : \{\mathbb{HOLD}, \mathbb{CHANGE}\} \\ a_2 : \{\mathbb{HOLD}, \mathbb{CHANGE}\} \\ a_3 : \{\mathbb{HOLD}, \mathbb{CHANGE}\} \\ a_4 : \{\mathbb{HOLD}, \mathbb{CHANGE}\} \end{bmatrix} \tag{2}$$

Rewards: As a common agreement,the goal of traffic control is to minimize the total time spent T in the traffic network over a time horizon T[6].

$$T_s = \delta t \sum_{t=0}^{T} N(t) \tag{3}$$

Due to conservation of vehicles we have

$$N(t) = N(t-1) + \delta t[d(t) - s(t)] \tag{4}$$

where $d(t)$ is traffic network demanding at time t and $s(t)$ is number of served vehicles at time t Hence,

$$N(t) = N(0) + \delta t \sum_{\tau=0}^{t-1}[d(\tau) - s(\tau)] \tag{5}$$

Substituting 3 in 1 We obtain

$$T_s = \delta t \sum_{t=0}^{K}[N(0) + T\sum_{\tau=0}^{t-1} d(\tau) - \delta t \sum_{\tau=0}^{t-1} s(\tau)] \tag{6}$$

The first two terms in the outer sum of 4 are independent of the control measures taken in the network; hence, minimization of T_s is equivalent to maximization of the following quantity:

$$S = \delta t^2 \sum_{t=0}^{T}\sum_{\tau=0}^{t-1} s(\tau) = \delta t^2 \sum_{t=0}^{T-1}(T-t)s(t) \tag{7}$$

Transition: Since number of cars ramping on the network is randomly generated, the state of next time step is uncertain. However, changes of traffic light is deterministic and it will effect the later states somehow. Therefore, we can use simulator to predict future state in a short term.

Initial State: The initial state will be randomly generated through Cell Transmission Model controller and random traffic lights settings.

3 Cell Transmission Model

Cell Transmission Model[4] can be treat as an discrete version of LWR model[3], which follows hydro-dynamic conservative theory. Car flow movement is based

on three constrains: largest density of each cell(N_i), largest flow rate(Q_i) and current cell occupation of target cell. In equation:

$$y_i(t) = \min\{n_{i-1}(t), Q_i(t), (w/v)[N_i(t) - n_i(t)]\} \tag{8}$$

where v is freeway velocity, w is back propagation speed, $n_i(t)$ means number of car in cell i at time t, $y_i(t)$ is input flow at time t for cell i .
So, the next state of movement will be:

$$n_i(t+1) = n_i(t) + y_i(t) - y_{i+1}(t) \tag{9}$$

We can address this time-space updates as a link structure as shown in following figure. Each node corresponds one cell, which has its largest density and current occupation. The edge between nodes describes the flow out through from parent cell to child cell.

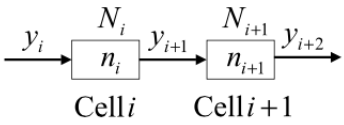

Figure 1: Cell Transmission Model

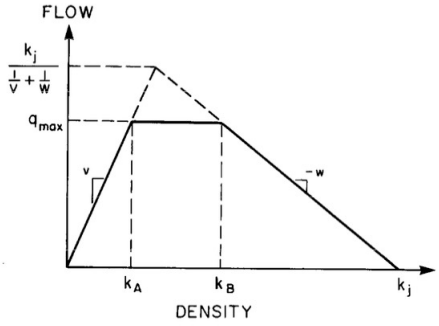

Fig. 4. Flow–density relationship for the generalized cell-transmission model.

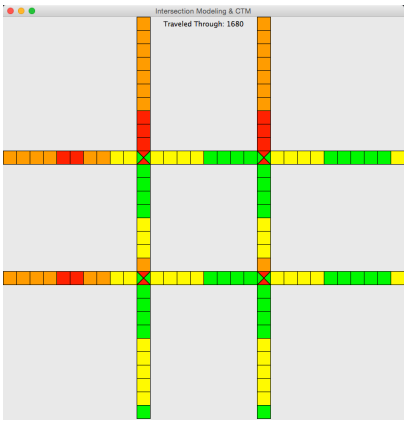

Figure 2: The left figure(a) shows triangle fundamental diagram of traffic flow and the right figure(b) is the corresponding intersection simulated through CTM

The above figure(b) shows the traffic condition under fix time control strategy. Color of cell describes density of that cell. Green means sparse, yellow means medium pressured, orange means quite crowded and red means in congestion. The intersection is expressed as one box with four triangles. the color of triangle represents traffic light(red to stop, and green to go). It is clear that, when traffic light is red, road will create a queue, and, when traffic light is green, the queue will gradually dissipate. And, we also can see the back propagated shockwave, when vehicles reach high density cells. Therefore, the traffic model works fairly ok for this project.

4 Monte Carlo Tree Search

The figure 3(a) is one simple example of the disadvantage of fix time control methods. We can see that horizon roads don't have much demanding, whereas vertical roads have very heavy load and even generating congestion. The fix-time controller doesn't notice the real-time environment changing and keeps doing the predesigned traffic light settings. To solve this problem, we need to consider real-time algorithms. Although there are lots of real-time methods in traffic control fields[8], MCTS seems to be a good technique to try on.

Monte Carlo Tree Search is one family of algorithm that has online view and good anytime behavior. We can continuously build a search tree until some predefined computational budget is reached(time, memory or iteration constraint). This property meets the demanding of traffic control field very well, especially for those busy intersections with large phrase interval(the period that light status is fixed).

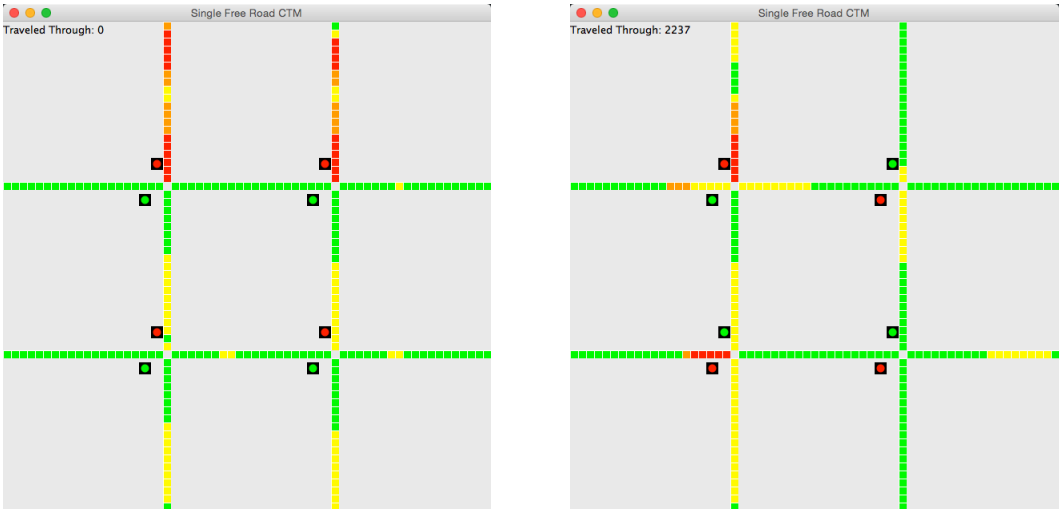

Figure 3: left figure shows the traffic network under fix-time traffic control strategy, while the right figure shows traffic network under UCT control

The Monte Carlo Tree Search keeps sampling the action trajectory in the computational budget based on some constrains such as tree policy and default policy. For each trail of sampling, it needs to run four functions step by step to update the tree.
1) Tree Search: start from the root node, search best action and transfer to

the child node through $P(s'|s,a)$ until reaching a node that doesn't fully expanded(some actions has not been tried).
2) Expand: Choose an action that not tried and apply state transition through $P(s'|s,a)$, add new state s' into the tree.
3) Simulation: input the new expanded state to the simulator and run default policy until reaching the terminate state
4) Back propagation: record the cumulative reward of one trajectory from root to leaf and back propagate this value to the root step by step.

UCT
Uniform sample method doesn't work good in limited budget of time, since it take too much time on bad options. To solve this problem, we can view each level of tree search as an bandit problem. Upper Confidence Bounds method is one of the methods to solve bandit problem. Using this method we can easily balance exploration-exploitation dilemma. So that the tree will always grows asymmetrically, but still occasionally update bad brunches of the tree. The UCB is shown as following equation.

$$\arg\max_{v' \in \text{Children of v}} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2\ln N(v)}{N(v')}} \tag{10}$$

Before applying UCT on urban traffic field, we should notice that the traffic condition keeps changing because of random number of cars driving into the network over time. And this causes regular UCT failure because it rarely be able to visit same state again, and, as consequence, the tree only grows horizontally. To capture this problem, we need a method to limit the sampling range to allow UCT visit same state.

In this project, I limited number of children of a state-action pair to 5, which means we only sample 5 times for one state-action pair and store the generated states as nodes. And, if the state action pair is visited more then 5 times, we only choose state from these 5 nodes.

Uniformly distributed child nodes are acceptable, but not rational. We can apply euclidean similarity method to address this. For each time we encounter fully expanded node, we sample new state through simulator. And, we compute euclidean distance of this sample with the five candidate child nodes. Then, the most similar candidate node will be chosen as the successive node of the trail.

The UCT methods requires a time period to do bunch of simulation before providing reliable recommendation of action. So, optimizing traffic light setting for current real-world state doesn't work well. We, therefore, use prediction-based traffic control optimization as shown in following figure. At the starting state of each decision making iteration, we simulate the state of the next decision making time point and grow an upper confidence tree with root of that predicted state(PS in the figure). Then the tree can grow until the real world state reach next decision making time point.

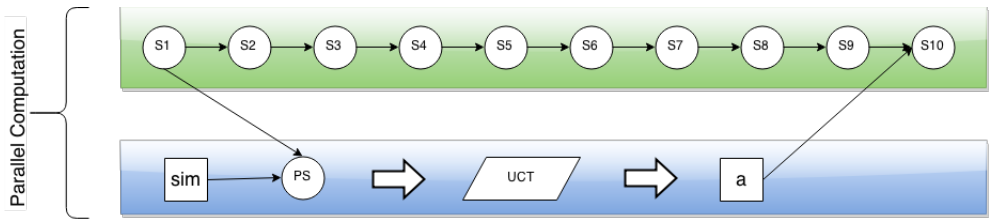

Figure 4: UCT traffic control

5 Experiments

To make sure my experiment is valid, in my experiment, I tested all the components of this implement separately, before running UCT algorithm on 'real world' traffic network. The first step is traffic flow test. This test randomly generates a traffic state based on given road settings, and it will record this traffic states for later reseting. Then, for each action, the simulator runs CTM and prints out the reward of that action.

Road Setting..Number of cars in each cell.

Road[0]:10,11,9,1,11,8,4,7,3,3,1,10,6,6,8,8,8,3,7,3
Road[1]:10,5,2,1,11,5,6,4,1,10,9,4,8,0,2,5,9,10,6,0
Road[2]:1,10,1,6,3,1,8,6,0,7,3,2,4,1,7,7,11,7,10,5
Road[3]:7,11,11,8,7,1,5,10,1,5,10,5,0,11,10,0,4,4,5,7
Road[4]:5,9,4,0,1,3,8,8,9,3,10,0,3,7,6,5,1,4,7,5
Road[5]:7,0,2,7,3,3,1,7,5,8,10,7,9,7,5,5,2,5,3,10
Road[6]:7,11,10,6,8,4,3,2,3,3,7,5,7,1,3,0,3,6,7,10
Road[7]:11,8,1,2,11,2,3,9,5,9,4,8,10,10,9,6,0,10,2,6
Road[8]:3,0,7,3,7,7,5,1,5,8,4,7,3,3,11,4,2,10,11,9
Road[9]:5,9,7,9,7,1,6,7,1,7,8,10,9,6,8,8,9,2,11,8
Road[10]:1,7,11,11,5,11,1,4,9,5,11,9,8,5,4,5,6,11,9,1
Road[11]:10,0,5,1,6,3,8,10,6,11,3,9,11,4,11,10,11,4,11,3

Rewards of action 0 is: 113
Rewards of action 1 is: 136
Rewards of action 2 is: 56
Rewards of action 3 is: 79
Rewards of action 4 is: 177
Rewards of action 5 is: 200
Rewards of action 6 is: 120
Rewards of action 7 is: 143
Rewards of action 8 is: 113
Rewards of action 9 is: 136
Rewards of action 10 is: 56
Rewards of action 11 is: 79
Rewards of action 12 is: 177
Rewards of action 13 is: 200
Rewards of action 14 is: 120
Rewards of action 15 is: 143

The action recommended by 1-level UCT:5

The final line shows that our implementation of UCT algorithm is correct. Note that 1-level UCT can cause the algorithm myopia, that is it can optimize reward for single step, but doesn't know the effect of the action for further decision making.

The following figure shows the performance comparison between two different control methods. The traffic network is designed to generate same demanding for both methods, which means the number of vehicles reach the network for both methods are same. The figure shows the number of vehicle passed through this network in 30 iterations for different control methods separately. Under MCTS control, the number of vehicle traveled through this network is much more than it under fix-time control.

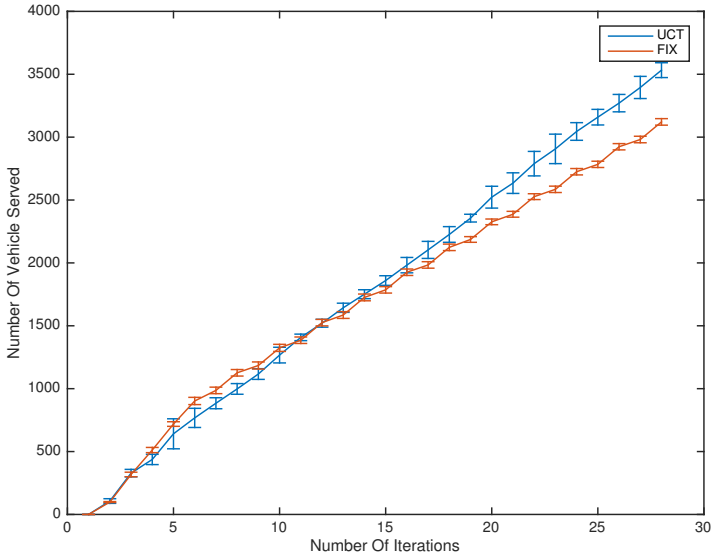

Figure 5: Performance of different control methods

This figure is averaged over 5 times of runs for each methods to make it smooth.

We can see that, from iteration 4 to iteration 11, the fix time control has better performance than MCTS. That is because the initial traffic condition is

randomly generated and every road has similar traffic load(similar number of cars on road). And, since fix time controller changes its traffic lights setting for each decision making point, it is quite fit the initial traffic demanding. However, as the time goes by, the traffic demand randomly changes, the performance of fix time control method gets worse.

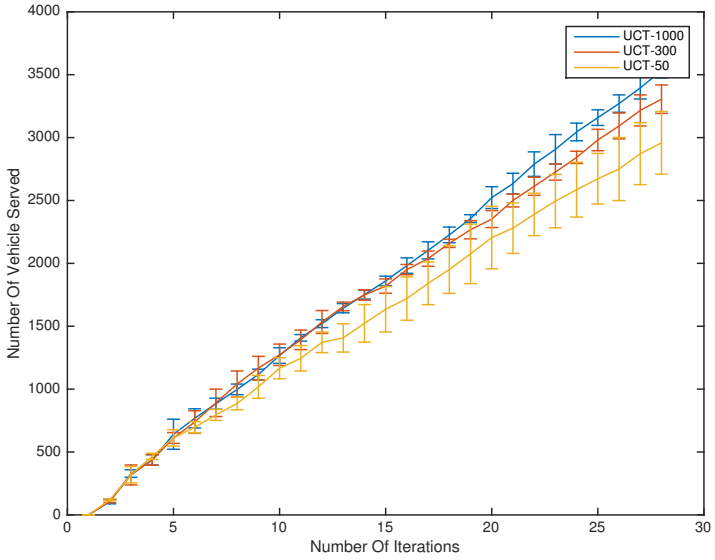

Figure 6: Performance of different time budget settings

Figure 6 shows the performance of UCT algorithm based on different time budget settings. We can see that the performance gets better when the budget is increased and the variance of the performance also gets smaller.

6 Conclusion

This work is to implement Upper Confidence Tree of Monte Carlo Tree Search family on urban traffic light control system. The motivation is that, traffic control is a complex markov decision processes problem, which is too complicate to learn complete transition function ore reward function, whereas Monte Carlo Planning has greet online view and this property will help to solve such complex problem. In this work, we implemented a reinforced UCT algorithm, which allows us apply this algorithm on continuous states environment. The algorithm has great any time behavior that makes sure it provide reliable recommendation of action in limited time budget.

The experiment results are quite near what we expected: the performance is better than fix-time control system. The challenge is to correctly implement traffic simulator and allows the UCT algorithm works on it. According to the experiment results, it is clear that the implementation is correct. We did test for each function of this implementation: State transition simulator test, Virtualization test, CTM model test, UCT implementation test and Combined test. Furthermore, we also modified the computational budget for UCT algorithm, it turns out the performance of this algorithm is tightly related to the computational budget. The more sampling we can do, the better the performance is.

The possible usage of this implementation is for those intersections which is still using fix time controller. Monte Carlo Tree Search method can response the environment changes on time, so that it can immediately optimize the out through of traffic flow.

Bibliography

[1] Ehsan Abbasnejad, Justin Domke, and Scott Sanner. Loss-calibrated monte carlo action selection. 2015.

[2] David Auger, Adrien Couetoux, and Olivier Teytaud. Continuous upper confidence trees with polynomial exploration–consistency. In *Machine Learning and Knowledge Discovery in Databases*, pages 194–209. Springer, 2013.

[3] Carlos Daganzo and CF Daganzo. *Fundamentals of transportation and traffic operations*, volume 30. Pergamon Oxford, 1997.

[4] Carlos F Daganzo. The cell transmission model, part ii: network traffic. *Transportation Research Part B: Methodological*, 29(2):79–93, 1995.

[5] Fern. Artificial intelligence course cs533: Alan fern. 2015.

[6] Arne Kesting. *Traffic Flow Dynamics: Data, Models and Simulation*. Springer Science & Business Media, 2012.

[7] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo–simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, 5(3 and 4):128–138, 2012.

[8] Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

[9] Stuart Russell. Artificial intelligence: A modern approach author: Stuart russell, peter norvig, publisher: Prentice hall pa. 2009.