TLab: Solution to KDD CUP 2021 City Brain Challenge

Fanyou Wu[*]
Purdue University
The organizerst Lafayette, Indiana, USA
wu1297@purdue.edu

Yang Liu[*]
Chalmers University of Technology
Gothenburg, Sweden
liuy@chalmers.se

Zhiyuan Liu
Southeast Uuniversity
Nanjing, China

Xiaobo Qu
Chalmers University of Technology
Gothenburg, Sweden

ABSTRACT

Traffic signals coordinate the intersection's traffic movements, and an intelligent traffic signal control algorithm is the key to transportation efficiency. Traffic signal control remains an active research topic because of the high complexity of the problem. The traffic situations are highly dynamic, thus requiring traffic signal plans to adjust to different conditions. In this paper, we propose a rule-based method to solve the problem and finally rank 10th for KDD CUP 2021 City Brain Challenge. The code is available at https://github.com/wufanyou/TLab-City-Brain-Challenge.

KEYWORDS

Traffic signals, signal control

1 INTRODUCTION

Traffic signals coordinate the intersection's traffic movements, and an intelligent traffic signal control algorithm is the key to transportation efficiency. Traffic signal control remains an active research topic because of the high complexity of the problem. The traffic situations are highly dynamic, thus requiring traffic signal plans to adjust to different conditions.

According to the transportation literature, max pressure (MP) control is one of the most advanced in traffic signal control [5]. The main goal behind MP is to reduce an intersection's "pressure", which may be informally described as the number of cars on incoming lanes minus the number of vehicles on departing lanes. MP has been shown to maximize the throughput of the whole road network by defining the goal as reducing junction pressure. However, MP's approach is greedy, resulting in locally optimum solutions.

Coordination may be done in traditional multi-intersection control by setting a fixed offset (i.e., the time gap between the commencement of green lights) among all junctions along an artery. [3]. In truth, it is a difficult undertaking since traffic from opposing directions cannot typically be accommodated at the same time. Some optimization-based approaches have been developed to address this issue. [1, 2] are developed to minimize vehicle travel time and/or the number of stops at multiple intersections. Instead of optimizing offsets, max pressure [4, 5] aims to maximize the throughput of the network so as to minimizing the travel time. However, these approaches still rely on assumptions to simplify the traffic condition and do not guarantee optimal results in the real world.

2 PROBLEM

Traffic signals coordinate the traffic movements at the intersection and a smart traffic signal coordination algorithm is the key to transportation efficiency. For a four-leg intersection (see figure below), 1 of 8 types of signal phases can be selected each period of time step, serving a pair of non-conflict traffic movements (e.g., phase-1 gives right-of-way for left-turn traffic from northern and southern approaches). In this competition, participants need to develop a model to select traffic signal phases at intersections of a road network to improve road network traffic performance.

In the final phase, a city-scale road network sample traffic data is provided. The organizer use exactly the same road network but

[*]Both authors contributed equally to this research.

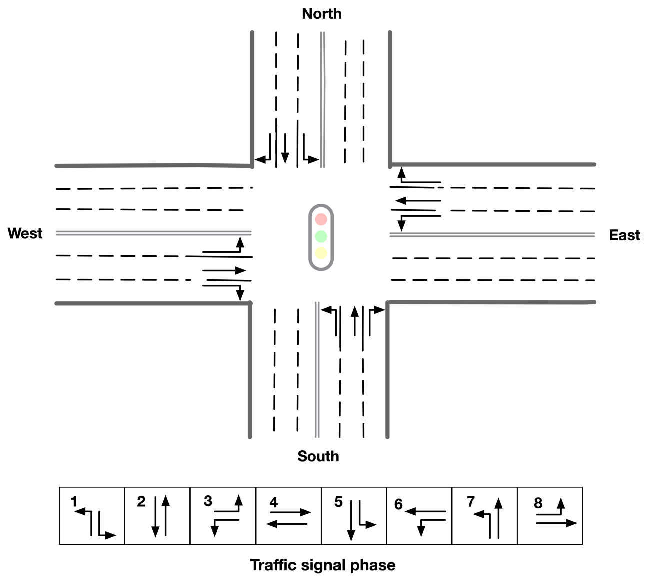


Figure 1: An illustration of the intersection used in this challenge.

different traffic data for scoring your submissions. Participants are encouraged to use the python script to generate your own sample traffic data for training and testing since the traffic settings for evaluation is not revealed.

3 EVALUATION

Total number of vehicles served (i.e., total number of vehicles entering the network) and delay index will be computed every 120 seconds to evaluate your submissions. The evaluation process will be terminated once the delay index reaches the predefined threshold 1.40.

The trip delay index is computed as actual travel time divided by travel time at free-flow speed. For an uncompleted trip, the free-flow speed is used to estimate the travel time of rest of the trip. The delay index is computed as average trip delay index over all vehicles served:

$$D = \frac{1}{N}\sum_{i=1}^{N} D_i \tag{1}$$

The trip delay D_i of vehicle i is defined as

$$D_i = \frac{TT_i + TT_i^r}{TT_i^f} \tag{2}$$

where TT_i is travel time of vehicle i, TT_i^r is the rest of trip travel time, estimated with free-flow speed and TT_i^f is the full trip travel time at free-flow speed.

The organizer will evaluate your solution on multiple traffic flow settings. The total number of served vehicles is computed over all evaluation scenarios. The overall delay index is computed as the average delay index among all vehicles of all scenarios.

The submission scoring and ranking process follows three principles:

- Solutions that served more vehicles will rank higher.
- If two solutions served the same number of vehicles, the one with lower delay index will rank higher.
- If two solutions served the same number of vehicles with same delay index, the one submitted earlier will rank higher.

4 ENVIRONMENT

CBEngine is a microscopic traffic simulation engine that can support city-scale road network traffic simulation. CBEngine can support fast simulation of road network traffic with thousands of intersections and hundreds of thousands of vehicles.

4.1 Road network data

Road dataset consists information about road segments in the network. In general, there are two directions on each road segment (i.e., dir1 and dir2). A snippet of road dataset is shown as follows.

Intersection data consists of identification, location and traffic signal installation information about each intersection. A snippet of intersection dataset is shown below.

This dataset describes the connectivity between intersection and road segments. Note that, we assume that each intersection has no more than four approaches. The exiting approaches 1 to 4 starting from the northern one and rotating in clockwise direction. Here, -1 indicates that the corresponding approach is missing, which generally indicates a three-leg intersection.

4.2 Flow data

low file is composed by flows. Each flow is represented as a tuple (start time, end time, vehicle interval, route), which means from start time to end time, there will be a vehicle with route every vehicle interval seconds. The format of flows contains several parts:

- The first row of flow file is n, which means the number of flow.
- The following 3n rows indicating configuration of each flow. Each flow have 3 configuration lines.
  - The first row consists of start time, end time and vehicle interval.
  - The second row is the number of road segments of route for this flow : k.
  - The third row describes the route of this flow. Here flow's route is defined by roads not intersections.

5 SOLUTION

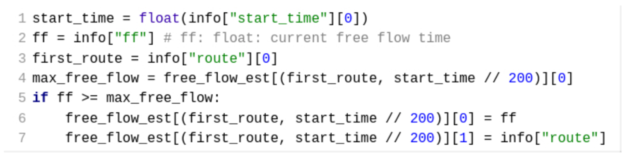
In this section, we will introduce our solution to this problem. In general, we implemented a rule-based method, which aims to maximize the flow movement during each time step for each intersection. Let s_l as the score at intersection lane l, then the next choice of the phase of intersection i is defined as:

$$p_i = \underset{k}{\operatorname{argmax}} \sum_{l \in L_{k,i}} s_l \tag{3}$$

where k is the possible signal phase and $L_{k,i}$ are permissible lanes of intersection i when the phase is set to k. The design of s_l then becomes the critical factor that affects performance. In this challenge, we designed this score based on number of vehicles of its current delay index. and also, we also took congestion issue into consideration. In the following section, we will introduce how to obtain s_l.

5.1 Estimation of current delay index

During the warm-up and qualification rounds, we could obtain free flow time TT_i^f of vehicle i from the environment. However, in the final round, free-flow time is not provided. This makes our previous solution lose its competitiveness due to its heavy dependence on free-flow time. Since the evaluation matrix delay index D_i is dependent on TT_i^f, it is necessary to have an estimation of TT_i^f. In section 4.2, we introduced the structure of the flow file and found that in the relatively short period (e.g., 200 second or equivalent 20 time step), the flow is repeatable. So it becomes possible to estimated the lower bound of TT_i^f. Figure 2 is the code snapshot which aims at estimate the lower bound of TT_i^f. In general, the late the vehicle departure in the same period, the more accurate the estimation will be.



```python
start_time = float(info["start_time"][0])
ff = info["ff"] # ff: float: current free flow time
first_route = info["route"][0]
max_free_flow = free_flow_est[(first_route, start_time // 200)][0]
if ff >= max_free_flow:
    free_flow_est[(first_route, start_time // 200)][0] = ff
    free_flow_est[(first_route, start_time // 200)][1] = info["route"]
```

Figure 2: Code snapshot of TT_i^f estimation

Here we use TT_i^e as the lower bound estimation of TT_i^f. Then we obtain each vehicle score:

$$v_i = min(\frac{t_c - t_i^s + TT_i^e - TT_i}{TT_i^e}, 10) \tag{4}$$

where t_c and t_i^s are the current time and the start time of vehicle i. We finally clip the v_i since the v_i will become extremely large when it is just departure.

5.2 Identification of permissible vehicle

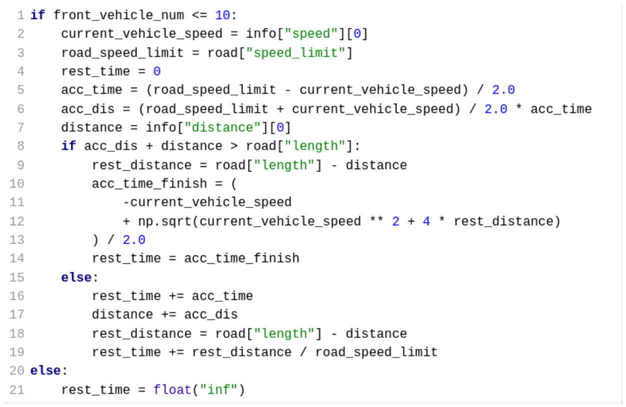
To identify the real permissible vehicles which could pass in the next time step, we need to investigate the real implementation of how the vehicle is speed up. We follow the example code and estimate the permissible vehicle use the code in Figure 3 to obtain permissible vehicles. After that, we can compute:

$$s_l = \sum_{i \in V_l} v_i w_i \tag{5}$$

where V_l is the permissible vehicle set for lane l and w_i is another parameter which controls the s_l.

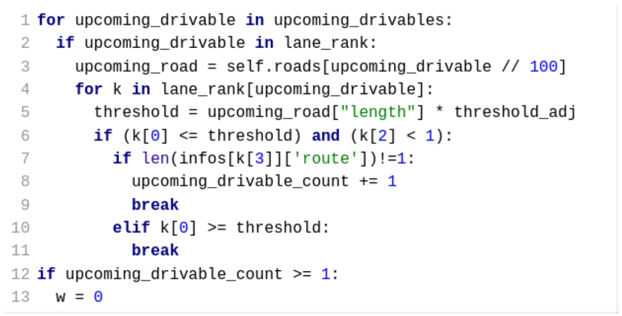
5.3 Detection of Congestion

The major drawback of the above method is that it does not consider the congestion issue of the upcoming lanes. We use the code in



```python
if front_vehicle_num <= 10:
    current_vehicle_speed = info["speed"][0]
    road_speed_limit = road["speed_limit"]
    rest_time = 0
    acc_time = (road_speed_limit - current_vehicle_speed) / 2.0
    acc_dis = (road_speed_limit + current_vehicle_speed) / 2.0 * acc_time
    distance = info["distance"][0]
    if acc_dis + distance > road["length"]:
        rest_distance = road["length"] - distance
        acc_time_finish = (
            -current_vehicle_speed
            + np.sqrt(current_vehicle_speed ** 2 + 4 * rest_distance)
        ) / 2.0
        rest_time = acc_time_finish
    else:
        rest_time += acc_time
        distance += acc_dis
        rest_distance = road["length"] - distance
        rest_time += rest_distance / road_speed_limit
else:
    rest_time = float("inf")
```

Figure 3: Code snapshot of permissible vehicle



```python
for upcoming_drivable in upcoming_drivables:
  if upcoming_drivable in lane_rank:
    upcoming_road = self.roads[upcoming_drivable // 100]
    for k in lane_rank[upcoming_drivable]:
      threshold = upcoming_road["length"] * threshold_adj
      if (k[0] <= threshold) and (k[2] < 1):
        if len(infos[k[3]]['route'])!=1:
          upcoming_drivable_count += 1
          break
        elif k[0] >= threshold:
          break
if upcoming_drivable_count >= 1:
  w = 0
```

Figure 4: Code snapshot of congestion detection

Figure 4 to detect the congestion. Once the congestion is detected, w_i will be set to 0 for vehicle i.

REFERENCES

[1] John DC Little, Mark D Kelson, and Nathan H Gartner. 1981. MAXBAND: A versatile program for setting signals on arteries and triangular networks. (1981).

[2] Dennis I Robertson. 1969. TRANSYT: a traffic network study tool. (1969).

[3] Thomas Urbanik, Alison Tanaka, Bailey Lozner, Eric Lindstrom, Kevin Lee, Shaun Quayle, Scott Beaird, Shing Tsoi, Paul Ryus, Doug Gettman, et al. 2015. *Signal timing manual*. Vol. 1. Transportation Research Board Washington, DC.

[4] Pravin Varaiya. 2013. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies* 36 (2013), 177–195.

[5] Pravin Varaiya. 2013. The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*. Springer, 27–66.

ABOUT US

Fanyou Wu is a Ph.D. candidate in Forestry and Natural Resources Department, Purdue University, West Lafayette, USA. He has gained a wealth of experience in the theory of interdisciplinary applications of machine learning techniques and has won many championships in AI competitions organized by leading international AI conferences or research institutes, including the championship of JDD (2019), the championship of IJCAI-Adversarial AI Challenge (2019), the championship of KDD Cup (2020), the second place in NeurIPS 2020 Traffic4cast Competition (2020) and the second place in CVRP 2021 The 2nd Agriculture-Vision Prize Challenge (2021).

Yang Liu earned his Ph.D. degree in Transportation Engineering from the School of Transportation at Southeast University, Nanjing, China. Now, he is a postdoctoral fellow in the Chalmers University of Technology, funded by Marie Skłodowska-Curie Actions. He has won three championships of Alibaba's Tianchi Algorithm Competition, the championship of IJCAI-Adversarial AI Challenge (2019), second place in the NeurIPS 2020 Traffic4cast Competition (2020) and the second place in CVRP 2021 The 2nd Agriculture-Vision Prize Challenge (2021).