Traffic Light Control Solution Based On Lane Score and Jam Rules

Chuan Qin
4Paradigm Inc.
Haidian, Beijing, China
qinchuan100@126.com

ABSTRACT

Traffic signal control is a challenging problem with high complexity. In this competition, the phases in each intersection are controlled by a policy every 10 seconds. Intuitively, the vehicles on the most congested lane should have the highest priority to pass. However, this report proposes that the congestion of the adjacent intersections is also an important factor to be considered in a control policy. Specifically, in this report, multiple rules are proposed to calculate the benefit of allowing the vehicles on a lane to pass(i.e., current lane score). In addition, to identify whether the next lanes in an adjacent intersection are congested, two jam rules are proposed. Based on the congestion on the next lanes, a jam penalty is calculated using sixteen rules (i.e., jam penalty of the next lanes). Finally, a phase score is derived. The phase to be allowed to pass in the next 10 seconds is the one with the highest phase score.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; Control methods; • **Applied computing** → Transportation.

KEYWORDS

Traffic Light Control, Lane Score, Jam Rules

ACM Reference Format:
Chuan Qin. 2018. Traffic Light Control Solution Based On Lane Score and Jam Rules. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In this competition, the trip delay index is used to evaluate the result. This 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 the rest of the trip. The delay index is computed as the average trip delay index over all vehicles served:

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

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

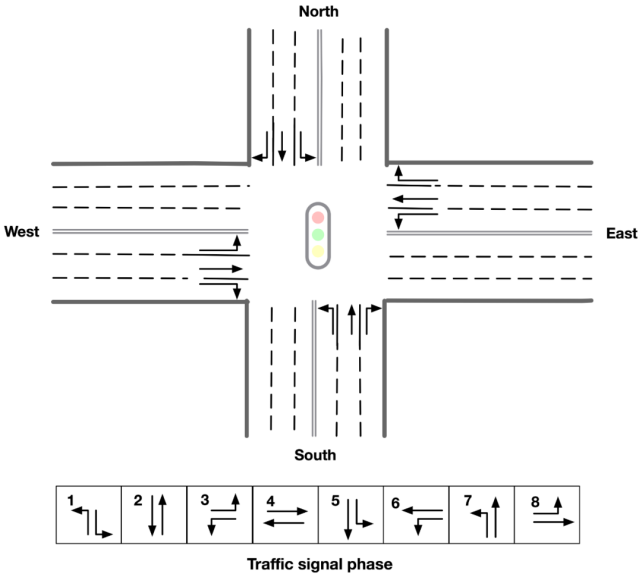

Figure 1: lanes and phases

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

As shown in Figure 1, there are 24 lanes and 8 phases at one intersection. Each phase serves a pair of non-conflict traffic movements. A score can be calculated for every lane and the phase score can be the sum scores of its lanes. Figure 2 shows the framework of the lane score calculation.

2 WORKFLOW

Algorithm 1 shows the main workflow, in which w = [0.43496, 0.44442, 0.56298, 0.57838, 0.521, 0.00024, 0.00495, 0.04571, 0.01381, 0.6427, 0.3, 0.33715, 0.4529, 0.2536, 0.2013, 0.1224, 0.0949, 0.05, 0.05, 0.1678, 0.1036, 0.0759, 0.07357, 0.1232, 0.028356, 0.05]. The phase with maximum score of every intersection is the decision for the next 10 second.

3 SCORES CALCULATION

3.1 Preparation In Advance

Firstly, the information from observation is processed to obtain $info\_for\_agent$, which records the vehicles information of each

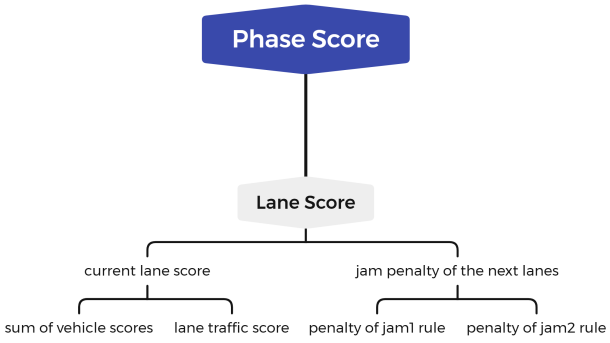

Figure 2: Framework

Algorithm 1 Main Workflow

```
info_lane, info_phase, actions
for phase = 1 to 8 do
  for lane in phase_lanes do
    lane score S=sum(w[i]*score_i) i=0,1,...,9
    info_lane[lane][phase] = info_lane[lane][phase]+S
    info_phase[agent][phase] = info_phase[agent][phase]+S
  end for
end for
for agent in agent_list do
  for phase = 1 to 8 do
    for lane in phase_lanes do
      for i = 1 to 16 do
        rule = jam_rules[i]
        if rule is True then
          info_phase[agent][phase] =
          info_phase[agent][phase]-
          w[i+9]*max(info_lane[lane])
        end if
      end for
    end for
  end for
end for
actions[agent] = argmax(info_phase[agent])+1
if max(info_phase[agent])==info_phase[agent][now_phase]
then
  actions[agent]=now_phase
end if
return actions
```

lane, containing vehicle id, speed, distance to the start point and current road id. Besides, the vehicles information list is sorted by distance for every lane so that the first element is the information of the first vehicle in the lane.

Secondly, the vehicle travel distance is calculated by the Algorithm 2. The vehicle accelerates from the current speed to the limit speed of the road by an acceleration of 2 and then keeps running at the limit speed of the road. The parameter *alpha* is a correction corresponding to the random deceleration.

Algorithm 2 Vehicle Travel Distance

```
Function name: cal_distance
Input speed, speed_limit, t, max_dist_per_second=[10000,10000,...,10000]

res=0, alpha=1, t_copy=t
if speed<speed_limit then
  alpha=0.8
end if
if speed==0 then
  t=t-1
end if
while t>0 do
  speed = min(speed+2, speed_limit)
  res = min(max_dist_per_second[t_copy-t],res+speed)
  speed = min(speed, max_dist_per_second[t_copy-t]-res)
  t = t-1
end while
return res*alpha
```

Finally, the time cost to travel a distance is calculated by Algorithm 3.

Algorithm 3 Vehicle Travel Time

```
Function name: cal_time
Input speed, speed_limit, dist
res=0
if dist==0 then
  return res
end if
if speed==0 then
  res = res + 1
end if
while dist>0 do
  speed = min(speed+2, speed_limit)
  dist = dist-speed
  res = res+1
  if res>20 then
    return res
  end if
end while
return res
```

3.2 Vehicle Scores

Vehicle score describes the benefit a vehicle could gain from passing the current lane. A larger vehicle score represents a larger benefit the vehicle gains from passing the current lane. For each vehicle, eight types of scores are calculated to capture this benefit.

- The first type of score ($score_0$) is calculated using the distance a vehicle could travel after passing the current road in 10 seconds divided by the limit speed of the road. In following equations, *speed* denotes current vehicle speed, *speed*0 means the speed of the first vehicle on this lane, *speed_limit* denotes the speed limit of the road, *length* is the road length,

distance denotes the distance from this vehicle to the start point of the current road.

$$free\_dist = length - distance \tag{3}$$

$$speed\_m = (speed + speed0)/2 \tag{4}$$

$$score_0 = max(0, \frac{cal\_distance(speed\_m, speed\_limit, 10) - free\_dist}{speed\_limit}) \tag{5}$$

- The second type of score ($score_1$) is similar to $score_0$ except that $score_1$ considers phase change. In following equations, *phase* denotes decision phase. *now_phase* denotes the phase before this decision.

$$dist\_5 = cal\_distance(speed\_m, speed\_limit, 5) \tag{6}$$

$$score_1 = \begin{cases} max(0, \frac{cal\_distance(speed\_m, speed\_limit, 10) - free\_dist}{speed\_limit}) \\ if\ phase == now\_phase\ or\ dist\_5 <= free\_dist \\ max(0, \frac{cal\_distance(0, speed\_limit, 5) + free\_dist}{speed\_limit}) \quad otherwise \end{cases} \tag{7}$$

- The third type of score ($score_2$) is similar to $score_0$ except that $score_2$ considers the maximum distance a vehicle could travel per second. In the Algorithm 4, $dist0$ denotes the distance of the first vehicle, *rank* denotes the order number starts from the first vehicle.

Algorithm 4 $score_2$ Calculation

```
max_dist_per_second=[]
speed_m=(speed+speed0)/2
free_dist=length-distance
for i = 1 to 11 do
  dist0 = dist0+speed0
  max_dist_per_second.append(dist0-5*rank-distance)
  speed0 = min(speed_limit, speed0+2)
end for
dist=cal_distance(speed_m, speed_limit, 10,
max_dist_per_second)
return max(0, dist-free_dist)/speed_limit
```

- The fourth type of score ($score_3$) is similar to $score_0$ except that $score_3$ considers phase change and the maximum distance a vehicle could travel per second.
- The fifth type of score ($score_4$) is similar to $score_0$ except that $score_4$ is calculated using the distance a vehicle could travel after passing the current road in 5 seconds divided by the limit speed of the road.
- The sixth type of score ($score_5$) is similar to $score_0$ except that $score_5$ is calculated using the distance a vehicle could travel after passing the current road in 15 seconds divided by the limit speed of the road.
- The seventh type of score ($score_6$) calculates the time a vehicle could travel after passing the current lane. In addition, this score considers phase change.

$$t = cal\_time(speed\_m, speed\_limit, free\_dist) \tag{8}$$

$$score_6 = \begin{cases} max(0, 10 - t) & if\ phase == now\_phase\ or\ (t < 10\ and\ t > 5) \\ t & if\ phase! = now\_phase\ and\ (t <= 5\ and\ t >= 0) \end{cases} \tag{9}$$

- The eighth type of score ($score_7$) is similar to $score_6$ except that it doesn't consider phase change.

$$score_7 = max(0, 10 - t) \tag{10}$$

3.3 Lane Scores

In addition to the sum of vehicle scores, another two types of lane scores ($score_8$ and $score_9$) are calculated by the following equations.

$$N = \sum_{i=1}^{n} b_i \tag{11}$$

$$b_i = \begin{cases} 1 & if\ distance > length - 10 * speed\_limit \\ 0 & otherwise \end{cases} \tag{12}$$

$$T = \begin{cases} 10 & if\ phase == now\_phase \\ 5 & otherwise \end{cases} \tag{13}$$

$$score_8 = \frac{cal\_distance(speed0, speed\_limit, T) * N * 5}{10 * speed\_limit * speed\_limit} \tag{14}$$

$$score_9 = \frac{N * 5}{speed\_limit} \tag{15}$$

4 JAM RULES

Two conditions are used to identify whether there is a traffic jam on a lane.

- `jam1`: The minimum vehicle speed of the lane is less than 2
- `jam2`:

$$\frac{vehicle\_number}{road\_length} > 0.048$$

The 16 jam rules used in the main workflow are as follows,

- $overload1 == 1$
- $overload1 == 1$ and $overload2 == 1$
- $overload1 > 1$
- $overload1 > 1$ and $overload2 == 1$
- $overload1 > 1$ and $overload2 > 1$
- $agent\_overload1$
- $agent\_overload1$ and $agent\_overload2$
- $overload1 > 0$ and $right\_overload1$
- $overload1 > 0$ and $right\_overload1$ and $right\_overload2$
- $overload1 == 0$ and $overload2 == 1$
- $overload1 == 0$ and $overload2 > 1$
- not $agent\_overload1$ and $agent\_overload2$
- $overload1 > 0$ and $overload2 > 0$ and $right\_overload1$ and $right\_overload2$
- $overload1 > 0$ and $overload2 > 0$ and $right\_overload1$ and not $right\_overload2$
- $overload1 > 0$ and $overload2 > 0$ and not $right\_overload1$ and $right\_overload2$
- among the intersections adjacent to the next intersection of the current lane, whether $agent\_overload1$ is true for at least one intersection

For each lane,

- `overload1`: denotes the number of lanes of the next road (except for the right-turn lane) that meet the jam1 condition.
- `overload2` : denotes the number of lanes of the next road (except for the right-turn lane) that meet the jam2 condition.
- `agent_overload1`: denotes whether all lanes except for the right-turn lanes in the next intersection meet the jam1 condition.
- `agent_overload2`: denotes whether all lanes except for the right-turn lanes in the next intersection meet the jam2 condition.
- `right_overload2`: denotes whether the right-turn lane of the next road meets the jam2 condition.
- `right_overload1`: To calculate *right_overload*1, I first identify the lanes that meet the jam1 condition on the next road (the left-turn lane or the straight lane, or both). Then, calculate the number of vehicles on the lane identified in the first step. If both lanes are identified in the first step, calculate the number of vehicles on the lane with fewer vehicles. Next, compare the number of vehicles in the right-turn lane on the next road with the vehicle number calculated in the second step. *right_overload*1 thus denotes whether the number of vehicles in the right-turn lane on the next road is larger than this vehicle number calculated in the second step.