深度学习导论

实验报告

KDD Cup CityBrain Challenge

PB18000149 吴越凡

2021 年 9 月 4 日

目录

插图

表格

1 Introduction

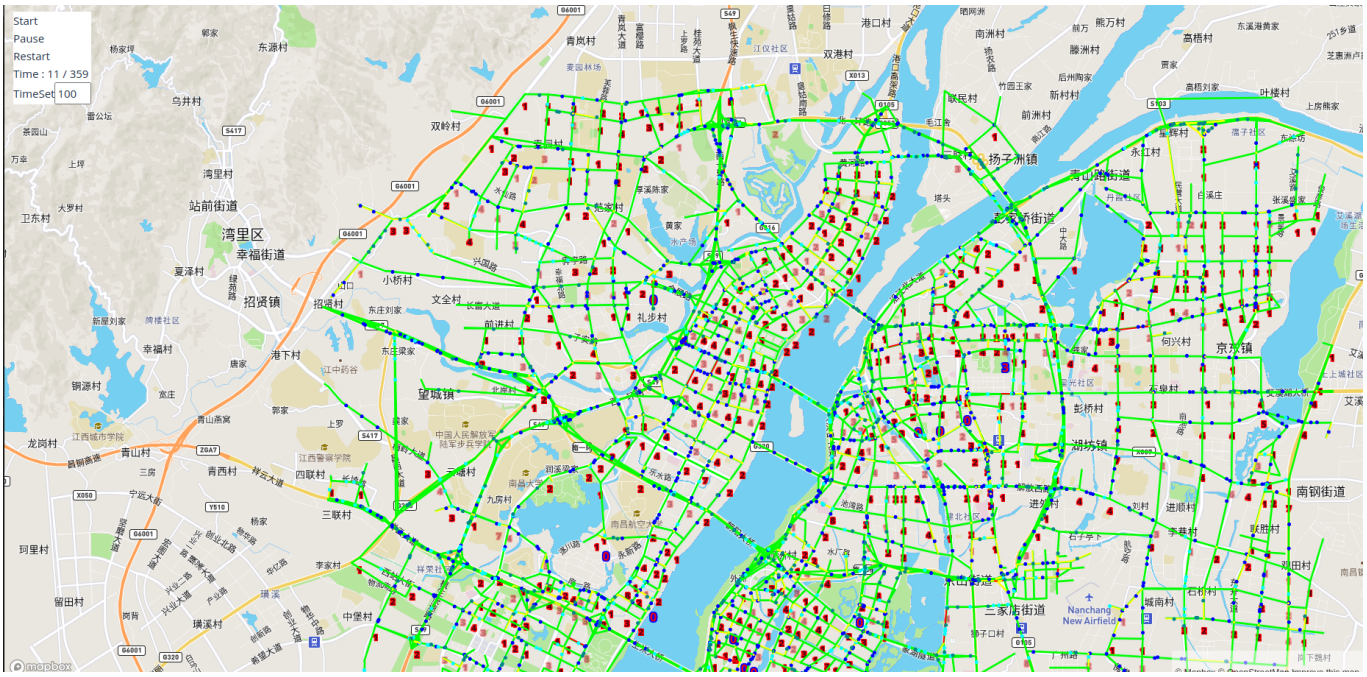

图 1: 模拟环境可视化

随着社会人口急剧增多，我们在生活中会遇到越来越多的堵车塞车情况。而值得我们进行思考的是为何会有城市拥有着类似的道路状况和人口情况，却可以在道路通勤方面存在着很大的差别。例如同为世界级大都市的纽约和东京，其中东京拥有更多的登记车辆，而东京信号灯交叉口仅仅比纽约市多 15%，远不及其登记车辆数的差异。纽约的道路是否已经进行了最大化利用，抑或是东京拥有着更好的通路规则?

在日常生活中，我们无法得知各个道路之中车辆的情况，因此我们无法对城市交通情况有一个宏观上的理解。导致我们没法去对于道路通勤情况进行有效的研究和探索。

在这次的 City Brain 比赛中，举办方提供了有效、可信的城市交通模拟平台，使得我们获得一定时间段上城市各道路交通、车辆情况，同时根据得到的信息去研究出更加高效的策略，进而去最优化城市交通堵塞情况，以期其在实际生活中能够提供一定的帮助和指导。

利用竞赛所提供的数据，我们可以把竞赛问题分裂为：根据实时的车流数据，我们去动态地调整交通信号灯的状态，希望在有限时间内使最大数量的车辆到达它们的目的地。并且在此基础上，我们希望能够获得更小的 delay，即实际到达目标时间与理论到达目标时间的差距。事实上，之前已有许多科研工作探究了这个问题。它们可以被分为两类：基于规则的算法（Rule Based Algorithm）与基于强化学习的算法（Reinforcement Learning Based Algorithm）。前者的策略常用于处理固定的交通信号操作策略。由于目前现实中很大一部分的交通信号灯仍是以固定频率变换信号的，这类算法依然具有一定的实际引用价值。然而，随着城市车辆数量的不断增加，城市人口的不断增多，现在的道路情况变得越来越复杂，所以动态调整信号灯的策略逐渐展现出更加旷阔的应用场景。动

态策略主要基于强化学习，它们能够根据实时交通状况的观察 (observation) 动态地调整交通信号灯的状态，从而宏观地将车流量从拥挤处向通畅处引导，最大化地挖掘城市交通的承载能力。当然，这类算法也面对着许多的困难，如环境的表示方法，车辆的建模方式等。研究者们仍然在不断探寻性能优越并在各种情况下鲁棒的策略。

在处理城市交通问题时，有一个很大的问题是模拟环境与真实环境的差距。在模拟环境中，会很容易对一些边界情况进行忽略和简化，比如超速或者极低速。而在这次的比赛中，我们的模拟环境是举办方对现实数据进行采集得到的结果。因此在这次的情况下，并不会出现 domain gap 的问题。

2 Motivation

没有人喜欢被困在城市交通中。虽然我们在城市里观察到很多车辆，但交通拥堵的原因仍然不清楚。是因为车辆的数量已经超过了城市的容量，还是我们没有最大限度地利用路网?

以世界上最大的两个城市为例。东京和纽约市的交通拥堵指数排名相似。然而，值得注意的是，东京的登记车辆比纽约市多 43%，而东京的信号交叉口仅比纽约市多 15%，道路长度仅比纽约市多 32%。(东京：313 万辆注册车辆，15,000 个交通信号灯，24,650 公里道路。纽约市：219 万辆注册车辆，13,000 个交通信号灯，18,684 公里道路。)

我们希望通过一个城市规模的路网及其来自真实交通数据的交通需求去构建出能够尽可能高效率的策略。

3 Environment

交通信号协调路口的交通运动，智能交通信号协调算法是提高交通效率的关键。对于四路交叉口（见下图），每个时间步长可以选择 8 种信号相位中的一种，为一对非冲突交通运动提供服务(例如，相位 1 为来自北部和南部入口的左转交通)。

4 Methods

4.1 Reinforcement Learning

在拿到这个比赛的赛题之时，大约在六月份左右，一开始看到问题时就在想这不就是一个很明显的强化学习问题吗，红绿灯作为 action，各个车道数量、车辆情况作为 obseration，然后最后的通过情况作为一个 reward。正如当时 github baseline 所开源的那样，强化学习的性能确实是优于简单的进行 random select 一个红绿灯情况的方法的。因此我们对强化学习，或者说该深度学习方法进行了认真的思考和探索，尽管在最后的结果上强化学习方法表现不佳。

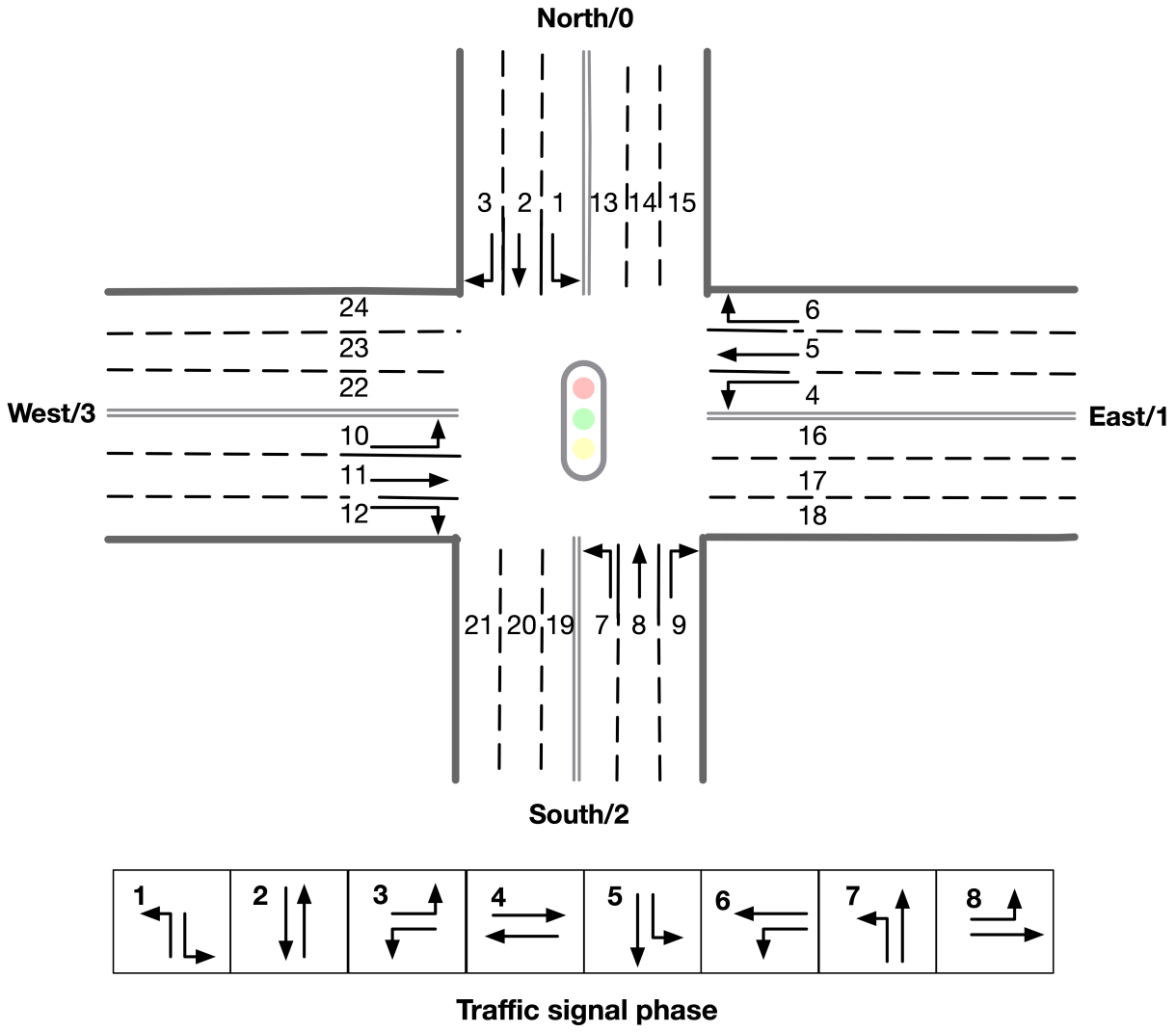

图 2: 道路索引与交通信号灯的八种状态

强化学习的方法包括三个部分：状态 (S:state)，动作 (A:action)，以及价值 (R:reward). 因此如果想要对强化学习方法进行改进，大致可以分为两个方面的改进：(1) 对状态函数，动作函数，以及价值函数进行改进. (2) 对训练方法进行改进. 如将 baseline 中的 DQN 变成 Policy Gradient, QR-DQN, PPO 等. 下面我们将先简单描述最简单的 RL 方法, 并逐一对我们所尝试的改进进行详细的阐述.

4.1.1 Basic DQN Method

该 Baseline 使用 Naive DQN 作为基本训练框架, 其中对于 state, action, reward 的定义如下:
(1) state: 每条车道上的车辆数据;
(2) action: 八种交通信号灯的状态;
(3) reward: next states' 的路口压力 (pressure).

定义了这些之后, 使用的 Q-network 为一个 hidden dimension=20 的两层 MLP, 便可以开始运行 DQN 算法.

4.1.2 针对 DQN 中 State,Action,Reward 等进行改动

1. 针对 State 进行修改

针对 State 的修改主要考虑以下的方面:

- 将速度加入到 State 信息里面。原先在 State 中仅仅包含了每一条道路上车辆的数量，再将每条道路上车辆的平均速度作为信息输入，从原先的 24->48

2. 针对 Action 进行修改

(1) 在原先的结果里面我们发现，并不是每一个路口都是设置成标准的四岔路口。通过之前的实验我们发现, 有一定概率会遇到三岔路口。而之前给定的红绿灯策略组合在进行选择的时候并没有将这一问题纳入考虑, 以致于会有一定的可能我们的策略会选择出出现死路的情况。这一类情况的发生会导致我们选择的策略有一部分失效，并不能达成车辆通行效率最大化。

针对三岔和四岔两种情况, 我们分别对其训练出一个网络进行预测。同时由于最少也是三岔路口, 所以对于每一个独立提取出来的三岔路口进行了旋转, 通过一系列的简单的旋转变换将其变为类似的三岔路口。在代码中我们选取朝东向的路口为缺失路口，可以减少一定的训练复杂度。这个方案被证明是有一定效果的。

(2) 此外在实验过程中发现，如果将路口的信息进行旋转，理应我们的预测结果也会根据对应的旋转进行对应的选择。但是在实验过程中发现，仅有大约 14% 的路口能够满足这种情况，仅仅高于 random select 的 12.5% 一点点，这并不满足我们的需要。所以考虑可能是因为训练数据不足，导致强化学习模型并不能到达一个期望的收敛点。所以分别对于路口进行旋转操作，90 度、180 度、270 度等，以期望通过一定程度的 data augmentation 去提高模型在模拟环境中的表现。

3. 对价值函数的改进。对于该价值函数, 我们认为需要将当前车道的长度考虑进去。借鉴于 [**?**], 我们将每条道路上的车辆除以道路上所能容纳最多车辆数目作为当前道路上的约化车辆数目。即:(p(l,m) 指从第 l 个路口到第 m 个路口的 pressure)

$$r = p(l, m) = \frac{x(l)}{x_{max}(l)} - \frac{x(m)}{x_{max}(m)} \tag{1}$$

其中 $x_{max}(l) = L(l)/M$。其中 $L(l)$ 是车道 l 的长度, M 指车与车之间的平均距离, 实验中。我们取的是 5 米。

4. 对于强化学习模型的更改在 DQN 的基础上，我们对于 DQN 变种（例如 QR-DQN，Rainbow DQN),Policy Gradient（PPO 等），FRAP 模型进行了尝试。

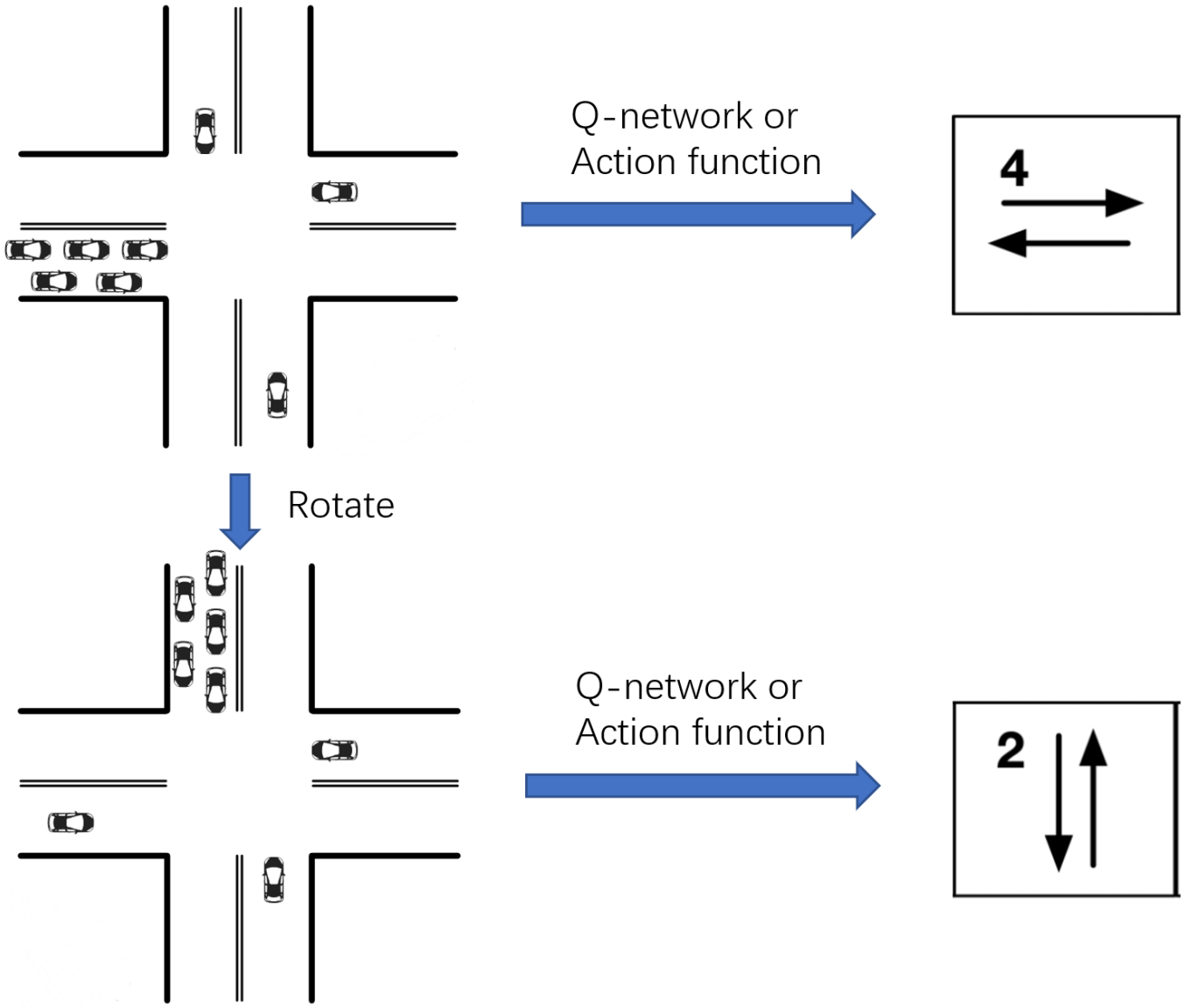

图 3: 假设该网络给出的预测结果为 4, 即左右通行。若我们将输入网络的状态进行旋转, 则网络预测结果也应该是原来的预测结果旋转后的结果, 即网络的输出应该变成 2。

4.2 基于规则的方法及贪心算法

在官方开源的第二套代码中, 有给出一个与深度学习无关的只用贪心算法的方法。其文件名为agent_MP.py。即每一步的选择为:

$$a_t = arg\max_{a_t in \mathcal{A}} \sum_{l \in In(a_t), m \in Out(a_t)} pressure(l, m) \tag{2}$$

其中 $pressure(l, m) = N_l - N_m$, N_l, N_m 分别是第 l 条车道和第 m 条车道上的车辆数目, $\mathcal{A}$ 为状态空间, 即八种信号灯的状态空间。

官方给的这个文件里面, 仅仅利用非常简短的代码就远远超过了强化学习模型能够达到的最好效果。也就是这个 baseline 的出现让我们意识到了可能强化学习并不是这一次 City Brain 问题的最佳策略选择。所以我们开始将代码研究重心转移到这一系列基于规则去进行简单决策的方法上去。

4.2.1 最终版

具体的规则如下:

- 首先计算出所有 phase(或者说 action) 的 waiting 车辆数目, 以及所有 phase 的 pressure, 其中 pressure 的算法如 Eq. (2) 相同;

- 其次对 waiting 的数量进行判断:
  (a) 如果最大车辆 phase 即为上次的 phase, 则进行如下选择:
  (1) 如果上次的 waiting 数量与现在的相同, 说明上游堵车不能缓解, 于是换到其他 phase:
  (i) 如果上次的 phase 对应的两个车道上游都有车辆排队, 则如果两条车道均为: 上游车道长度/上游堵车数目 < 5, 则观察这两个车道中哪个车道堵得更厉害, 并将和这个更堵的车道相关的 phase 全部从候选集合中删除;
  (ii) 如果上次的 phase 对应的车道有且只有一个有车辆排队, 则如果有车辆排队的车道上满足: 上游车辆长度/上游堵车数目 < 5, 则将和这个车道相关的 phase 删除, 在剩下的当中选择。
  (iii) 如果两个车道中都没有车辆排队, 则无需改变 phase。
  (2) 无需改变 phase。(虽然上次最堵的现在仍然最堵, 但是相比于上次有所缓解, 因此不需要改变状态)。
  (b) 最大排队 phase 并非上次的 phase, 则:
  (1) 如果上次的 phase 对应的排队车辆数目当前为零, 则直接设定为排队最长的 phase;
  (2) 如果上次的 phase 对应的排队车辆数目在当前最大排队数目的 30%~60% 之间, 且最大排队数目小于 8, 则 phase 不变。
  (3) 如果上次的 phase 对应的排队车辆数目大于最大排队数目的 60%, 且最大排队数目小于

8, 则 phase 不变。

(4) 否则, 直接选择最大排队数量的 phase。

一些其余的性能提升方法:

1. 平滑车辆数目 对每辆车来说, 从环境返回的 info 变量中可以读到如下信息: 每辆车的" 产生时间": t_0, 现在的时间 t, 以及该车预期到达目的地的时间 t_f。其中 t_f 代表着这辆车在所有的道路上都以该道路最大限速行驶到目的地所需要花费的时间。这辆车所在的道路 i, , 在当前车道上所行驶的距离 d, 以及这辆车从出发地点到结束地点所要经过的路径 $1, \cdots, n$。其中 $1 \leq i \leq n$。在给定这些信息之后, 计算 delay_index 的方式如下:

$$delay\_index = \frac{(t - t_0) + t_{lf}}{t_f} \tag{3}$$

其中 t_{lf} 为剩下 (left) 的路如果全部按照最大速度行驶所需要的时间, 具体计算方法为:

$$t_{lf} = \frac{l_i - d}{v_{im}} + \sum_{j=i+1}^{n} \frac{l_j}{v_{jm}} \tag{4}$$

其中 v_{jm} 为第 j 条道路上的最大限速, 其中 $1 \leq j \leq n$, l_j 表示第 j 条道路的长度, $1 \leq j \leq n$。

在这个基础上, 我们希望求出 delay_index 对时间的导数。这里先作一个思想变换, 将第 s 条道路限速换算成 v_{im}, 利用 $l'_s = v_{im} * t_s = v_{im} * \frac{l_s}{v_{sm}}$ 计算出约化的道路距离 l'_s, 在这样的变换下, t_f 将变成新的 t'_f:

$$t'_f = \sum_{s=1}^{n} l'_s / v_{im} = l_{all} / v_{im} = \sum_{i=1}^{n} l_s / v_{sm} = t_f \tag{5}$$

由此可知 $t_f = t'_f$, 但这样的一个变化使得得到一个新的理想情况: 所有道路的限速是一致的。这样将 Eq. (3) 对时间求导, 得到如下结果:

$$\frac{\partial delay\_index}{\partial t} = \frac{1}{t_f}(1 + \frac{\partial t_{lf}}{\partial t}) = \frac{1}{t_f}(1 + \frac{\partial l'_{lf}/\partial t}{v_{im}}) \tag{6}$$

其中 l'_{lf} 为剩下的约化路程: $l'_{lf} = l_i - d + \sum_{s=i+1}^{n} l'_s$。将该式代入 Eq. (6), 得到:

$$\frac{\partial delay\_index}{\partial t} = \frac{1}{t_f}(1 - \frac{1}{v_{im}}\frac{\partial d}{\partial t}) = \frac{1}{t_f}(1 - \frac{v_t}{v_{im}}) \tag{7}$$

其中 v_t 为当下车的速度。

在后面的计算当中, 我们便将该式右边的结果作为对一条车道上车辆数目估计的指标。我们计算出所有车辆的平均预期时间, 作为 $\bar{t}_f$, 作为一个常数乘到上面的式子中, 得到:

$$\frac{\partial delay\_index}{\partial t} = \frac{\bar{t}_f}{t_f}(1 - \frac{v_t}{v_{im}}) \tag{8}$$

在这些推导之后, 我们便 $\frac{\bar{t}_f}{t_f}(1 - \frac{v_t}{v_{im}})$ 作为平滑车辆数目的计算方式。从这个式子中可以看出, 如果一辆车停在原地, 即 $v = 0$, 则它将视为在这条车道上有 $\frac{\bar{t}_f}{t_f}$ 辆车。同理, 如果它已经达到最大速度, 或者已经超速, 我们便将这辆车对这条车道上车辆数目的贡献视为 0。

Methods	Max Support Vehicle Number	Delay Index
DQN + MLP(20dim)	70026	1.6215426
DQN + MLP(32dim)	70026	1.6215323
DQN + FRAP	41505	1.6125042
DQN + CoLight	35238	1.6524098
DQN + MLP(32 dim) + Rotation Augmentation	56312	1.6321188
DQN + 3inter + MLP(32 dim)	70026	1.6014494
QR-DQN + 3inter + MLP(32 dim)	77462	1.6201493
Rainbow + 3inter + MLP(32 dim)	77462	1.6120203
RBA(Rule Based Greedy Algorithm)	126572	1.6016482
RBA + 平滑车辆数目	126572	1.5739681
RBA + 考虑下游车道的容量	126572	1.5664674
ARGA(Advanced Rule based Greedy Algorithm)	**126572**	**1.5363224**
ARGA + 容量和排队车辆数的传递，1-hop	126572	1.6248363
ARGA + 容量和排队车辆数的传递，2-hop	117897	1.6031824
ARGA + ST	93352	1.6325314
ARGA + SQN	102008	1.6117023

表 1: 实验结果。其中 MLP(20dim) 和 MLP(32dim) 分别表示该两层 MLP 的 hidden dimension 分别为 20 维和 32 维。FRAP 和 CoLight 表示将 DQN 中的 Q-network 直接替换为 FRAP 和 CoLight 网络。Rotate Augmentation 表示使用旋转增加样本数据量。

2. 考虑下游车辆数量 在考虑车辆数量时，除了简单的考虑上游车辆的数量，下游车辆的数量也会一定程度上影响道路通勤速度。所以考虑将之前的 pressure 变为一个 capacity 和之前 pressure 取 min 的操作。

5 Experiments

5.1 各方法实验结果

表 1 中包含了在实验中的一系列结果。

5.2 提交记录

YUNQI ACADEMY OF ENGINEERING Home Organizer Submission Leaderboard Log in

Rank	Team Name	Total Served Vehicle	Delay	Submission Time
21	ECNU_MAIL	126572	1.5270918227170092	2021-06-10 19:29:57
22	ASD	126572	1.527190088349228	2021-06-10 19:36:58
23	NTT_DOCOMO_LABS	126572	1.5281032706781843	2021-06-10 19:53:41
24	CTSU	126572	1.5363224711996677	2021-06-10 00:22:42
25	LingyuZhu	126572	1.5363433780220879	2021-06-09 23:26:03
26	Intelligame	126572	1.5399923759520233	2021-06-09 22:44:12
27	Zhang	126572	1.5400006408372289	2021-06-07 17:52:14
28	IntelliGame_CB_Model	126572	1.5413206502025996	2021-06-09 22:36:42
29	Argus_Sec	126572	1.5438710972454817	2021-06-09 21:15:10
30	Infinity	126572	1.5460767193349874	2021-06-08 16:50:41

YUNQI ACADEMY OF ENGINEERING Home Organizer Submission Leaderboard iven

2	117897	1.6101588742065351	2021-06-10 19:54:52	Finished	iven
3	117897	1.609191520480371	2021-06-10 18:57:04	Finished	iven
4	117897	1.6205120704535927	2021-06-10 18:56:23	Finished	iven
5	126572	1.5363433780220879	2021-06-09 23:26:03	Finished	iven
6	126572	1.5419037561685216	2021-06-09 21:16:43	Finished	iven
7	126572	1.5451265485605281	2021-06-09 12:36:13	Finished	iven
8	126572	1.541968934979384	2021-06-09 12:22:33	Finished	iven
9			2021-06-08 23:43:35	Failed	iven
10	126572	1.5480712576692983	2021-06-08 23:28:58	Finished	iven

< 1 2 3 4 >

图 4: 我们的队伍名称为 LingyuZhu, 以吴越凡的 iven 用户名提交。

6 Conclusion

在本次的比赛中，我们对于城市中红绿灯策略进行了探索，并且针对这一系列问题进行了关于新颖的深度强化学习和传统的基于规则的策略方案。

在此次的比赛中，我们发现在强化学习中，进行合适参数的调整和选择，以及对于 Reward 函数等细节方面的调整尤为重要，合适的参数选择和 Reward 函数可以得到更好的结果。

此外我们也发现强化学习办法并不是想象中那么有效，在最后我们还是选择了较为简单的传统基于规则的办法，并且得到了较好的结果。从这次的比赛中可以得出，并没有一个一成不变并且最好的方法，我们只能够在给定的条件下进行最合适的方法选择并且进行优化。

7 Acknowledgement

竞赛内容主要与王禹、陈泽远组合作完成，报告内容参考了其模板及内容。