

**企业实习中期报告**

学号： 2016220303007

姓名： 李紫延

专业方向： 嵌入式系统

企业名称： 成都康赛信息技术有限公司

实习岗位名称： 算法工程师

企业指导教师： 陈科

院内代管教师： 吴佳

**信息与软件工程学院**

目 录

[1. 企业实习的进展情况 1](#_Toc479152473)

[1.1. 实习工作完成情况 1](#_Toc479152474)

[1.2. 知识技能学习情况 3](#_Toc479152475)

[1.3. 职业素养学习培养 3](#_Toc479152476)

[1.4. 工程协作与交流情况 3](#_Toc479152477)

[2. 存在问题与解决方案 3](#_Toc479152478)

[2.1. 存在的主要问题 3](#_Toc479152479)

[2.2. 解决方案与可行性研究 4](#_Toc479152480)

[3. 前期任务完成度与后续实施计划 5](#_Toc479152481)

[参考文献 6](#_Toc479152482)

说明:

1.报告要求6000字以上。

2.本模板仅为基本参考，请各位同学根据个人情况进行目录结构扩展。

3.封面和目录单面打印，从正文第1页开始双面打印。

4.打印时删除本段说明。

## 1. 企业实习的进展情况

### 1.1 实习工作完成情况

截至目前，通过前期的实验和分析总结，对于交通场景中交通信号灯和自动驾驶汽车的联合控制已经有了可行的解决方案，在和传统方法的对比实验中也取得了良好的结果。

针对交通场景中交通信号灯和自动驾驶汽车的联合控制这一问题，主要工程问题有以下几点：

（1）在SUMO环境中如何解决自动驾驶汽车驶出场景的问题；

（2）如何设计智能体(Agent)的模型以及智能体的网络结构；

（3）如何将环境(Environment)中的信息抽象成智能体的状态输入；

（4）如何定义智能体的输出(Action)和如何将智能体输出的动作执行到环境中；

（5）如何定义环境对智能体的反馈(Reward);

（6）如何解决交通规模变大导致智能体动作空间(Action Space)急剧增大的问题。

问题的详细描述与详细解决方案参见第二部分。对于以上问题的分析与解决贯穿了项目的整个过程，问题（1）属于强化学习中环境搭建的问题，问题（2）-（5）属于强化学习模型设计的问题，问题（6）属于强化学习可拓展性的问题，我们设计的方案的创新点与亮点主要体现在对第二部分和第三部分问题的解答。针对交通信号灯和自动驾驶汽车联合控制的问题，我们利用了强化学习中多智能体(Multi-agent)的方法，设计了两个独立的智能体来分别控制交通信号灯和自动驾驶汽车；针对动作空间随交通规模变大而急剧增大难以学习的问题，我们提出了一种分级(Hierarchical)的方法，利用预先设定好的底层控制器（Local Controller）作为交通灯或自动驾驶汽车的子策略(Sub-policies)，智能体的工作是协调各个子策略，作为主控制器(Master Controller)来控制交通信号灯和自动驾驶汽车。利用以上这样的方法，智能体将着眼于全局的协调控制而不是每一个独立单位自身的控制，动作空间大大减小；同时，使用先验的子策略更有助于策略的搜索，在提高训练效率的同时使训练更加稳定。智能体模型图参见图1-1。

在模型的训练方法上我们选择了使用了近端策略优化方法(Proximal Policy Optimization, PPO)。PPO算法是一种基于策略梯度(Policy Gradient)的离线(off-policy)算法，它能够利用旧策略(old policy)通过重要性采样(important sampling)在有约束的条件下高效的更新策略参数，具体算法的介绍参见1.2部分。在Agent的网络结构上我们使用了有两层隐藏层的全连接网络，每层隐藏层有512个单元，输出根据交通信号灯和自动驾驶汽车的特性进行了各自相应的调整，网络模型见图1-2。

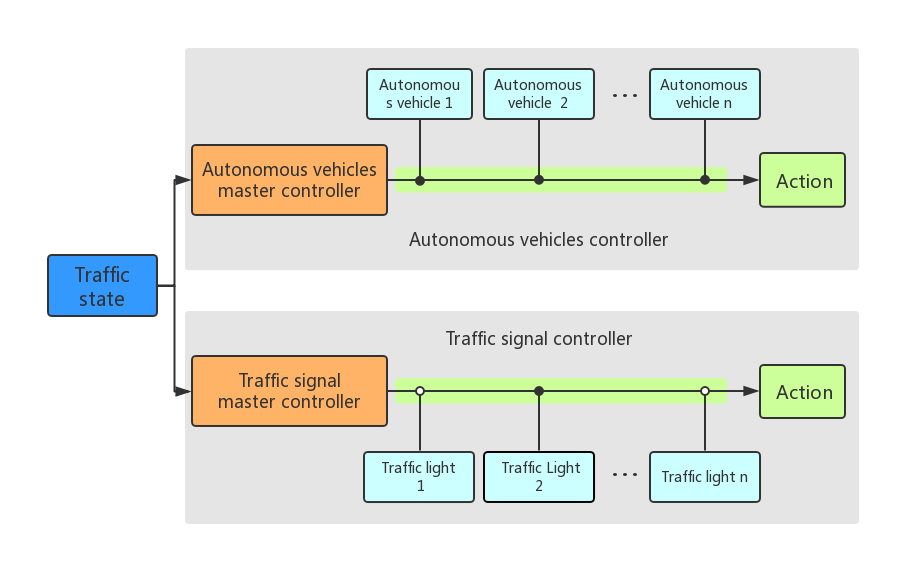


图1-1 分层联合控制模型结构图

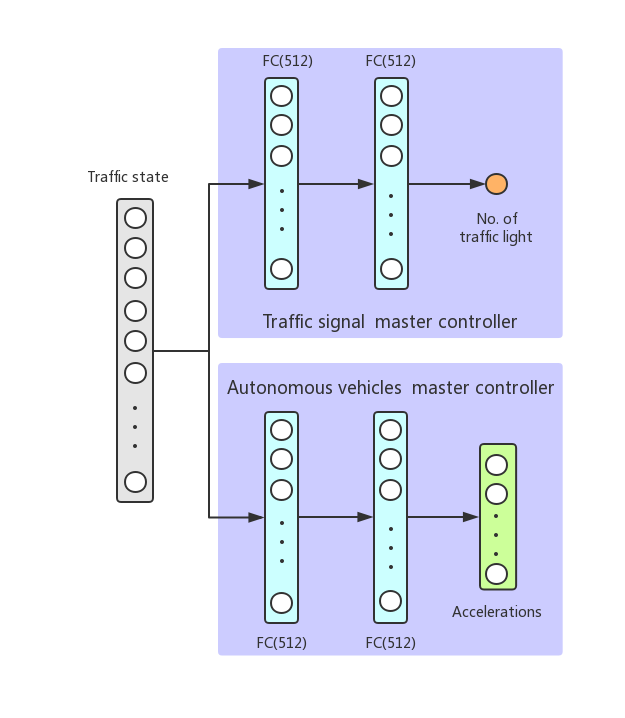


图1-2 智能体网络结构示意图

### 1.2 知识技能学习情况

在前期实习过程中对于知识框架的学习与应用比较宽泛与浅显，在近段时间的实习过程中学习的主要知识是如何使用Flow与SUMO来进行项目实践同时学习与更改源码以满足我们的个性化需求，同时在学习机器学习与强化学习的基础上学习一些先进的算法来进行我们的智能体的训练。

#### 1.2.1 Flow与SUMO的学习

Flow是一个将深度强化学习方法应用与混合交通网络的框架， SUMO(Simulation of Urban Mobility)是一个开源，微观，多模态的交通模拟软件。由于我们的模型是在Flow框架下调用SUMO来模拟交通交互，所以必须要对这两个工具熟练应用，能够熟练调用Flow提供的API来设计交通网络模型并能更改Flow源码来满足我们的需求。学习并熟练使用这两个工具是完成这一项目的基础，因为这正是强化学习中“环境”这一关键要素，我们需要通过环境来定义交通网络的类型、大小，定义智能体如何从环境中获取关键信息以及定义如何将智能体的动作应用于环境的交互。

对于以上两个工具的学习主要通过对于Demo的实现以及源码的分析解读完成，通过前期大量的实验以及对开源代码的分析理解，我们已经可以根据自己的需求来设计交通网络的大小长度以及交通灯灯的类型和汽车的类型与车流的大小等等。同时，我也学习了如何从环境中获取并处理信息来作为状态的输入（见代码1-1）以及奖励值（代码1-2）。

代码1-1 获取环境状态信息

|  |
| --- |
| def get\_state(self):  speeds = []  dist\_to\_intersec = []  edge\_number = []  quenes = []  velocity\_avg = []  max\_speed = self.speed\_limit  max\_dist = self.edge\_length  all\_observed\_ids = []  for node, edges in self.scenario.get\_node\_mapping():  for edge in edges:  observed\_ids = \  self.k\_closest\_to\_intersection(edge, self.num\_observed)  all\_observed\_ids += observed\_ids  # check which edges we have so we can always pad in the right  # positions  speeds += [  self.k.vehicle.get\_speed(veh\_id) / max\_speed  for veh\_id in observed\_ids  ]  dist\_to\_intersec += [  (self.edge\_length  - self.k.vehicle.get\_position(veh\_id)) / max\_dist  for veh\_id in observed\_ids  ]  edge\_number += \  [self.\_convert\_edge(self.k.vehicle.get\_edge(veh\_id))  / (self.k.scenario.network.num\_edges - 1)  for veh\_id in observed\_ids]  quene = 0  lane\_ids = self.k.vehicle.get\_ids\_by\_edge(edge)  lane\_speed = self.k.vehicle.get\_speed(lane\_ids)  if len(lane\_speed)>0:  velocity\_avg.append(np.mean(lane\_speed)/self.speed\_limit)  else:  velocity\_avg.append(0)  for spd in lane\_speed:  if spd<0.5:  quene += 1  quenes += [quene / self.num\_observed]    if len(observed\_ids) < self.num\_observed:  diff = self.num\_observed - len(observed\_ids)  speeds += [0] \* diff  dist\_to\_intersec += [0] \* diff  edge\_number += [0] \* diff  av\_ids = self.k.vehicle.get\_rl\_ids()  av\_speeds = []  av\_pos = []  av\_edge\_number = []  if len(av\_ids)>0:  av\_speeds = [self.k.vehicle.get\_speed(veh\_id) / self.speed\_limit  for veh\_id in av\_ids]  av\_pos = [self.k.vehicle.get\_position(veh\_id) / self.edge\_length  for veh\_id in av\_ids]  av\_edge\_number =[self.\_convert\_edge(self.k.vehicle.get\_edge(veh\_id)) / (self.k.scenario.network.num\_edges - 1)  for veh\_id in av\_ids]  if len(av\_ids) < self.num\_rl:  diff = self.num\_rl - len(av\_ids)  av\_speeds += [0] \* diff  av\_pos += [0] \* diff  av\_edge\_number += [0] \* diff  tl\_state = self.last\_change.copy()  for i in range(self.num\_traffic\_lights):  tl\_state[i, 0] = tl\_state[i, 0] / self.min\_switch\_time  tl\_state[i, 3] = tl\_state[i, 3] / (self.phase\_time[i, int(tl\_state[i, 1]+0.1)] + 1)    self.observed\_ids = all\_observed\_ids  state = np.array(  np.concatenate([  speeds, dist\_to\_intersec, edge\_number, quenes, velocity\_avg,  av\_speeds, av\_pos, av\_edge\_number,  tl\_state.flatten().tolist()  ]))  return {'av':state,'tl':state} |

代码1-1的主要功能是通过SUMO提供的接口来获取各种交通状态，包括交通信号灯的状态、所有车辆的位置、速度、每个路段车辆的平均速度排队数量等等，然后把这些数据做归一化处理然后打包成向量作为智能体的状态输入

代码1-2 从环境获取信息定义奖励值

|  |
| --- |
| def compute\_reward(self, rl\_actions, \*\*kwargs):  lead\_edge = {}  lead\_edge['av\_0'] = ['bot1\_0','bot1\_1','bot1\_2'] # TODO: Fixed  lead\_edge['av\_1'] = ['top1\_1','top1\_2','top1\_3']  lead\_edge['av\_2'] = ['left1\_1','left2\_1','left3\_1']  lead\_edge['av\_3'] = ['right0\_1','right1\_1','right2\_1']  # self.recent\_tl\_actions[self.step\_counter % 2] = rl\_actions['tl']  reward = {}  av\_reward = []  for i in range(self.num\_rl):  av\_id = 'av\_{}'.format(i)  lane\_vehs = []  for edge in lead\_edge[av\_id]:  lane\_vehs += self.k.vehicle.get\_ids\_by\_edge(edge)  # print(followers)  speeds = []  speeds = self.k.vehicle.get\_speed(lane\_vehs) if len(lane\_vehs)>0 else [self.speed\_limit]  # print(speeds)  mean\_speed = np.mean(speeds)  s = max(mean\_speed, 0)  av\_reward.append(s)  av\_sum\_action = np.mean(rl\_actions['av'])  av\_forward\_reward = 5 \* av\_sum\_action  reward['av'] = 0.8 \* np.mean(av\_reward) + av\_forward\_reward  # reward['av'] = rewards.average\_velocity(self)    tl\_reward = []  penalty = 0  for j in range(self.num\_traffic\_lights):  quenes = []    node = 'center{}'.format(j)  for node\_edge in self.scenario.get\_node\_mapping():  if node\_edge[0] == node:  for edge in node\_edge[1]:  quene = 0  veh = self.k.vehicle.get\_ids\_by\_edge(edge)  spd = self.k.vehicle.get\_speed(veh)  for s in spd:  if s<1:  quene+=1  quenes.append(max(quene,0))  tl\_reward.append(-max(quenes)\*\*1.2)  for i in range(self.num\_rl):  av\_id = 'av\_{}'.format(i)  if av\_id in self.k.vehicle.get\_rl\_ids():  speed = self.k.vehicle.get\_speed(av\_id)  pos = self.get\_distance\_to\_intersection(av\_id)  if pos < 30 and speed < 2:  penalty -= (30-pos) \* (self.speed\_limit - speed)  if self.last\_tl\_action == rl\_actions['tl']:  penalty -= 10    reward['tl'] = (np.sum(tl\_reward) + 0.1 \* penalty) / 4  if self.env\_params.evaluate:  if 'tl' in rl\_actions.keys():  reward['tl'] = rewards.min\_delay\_unscaled(self)  if 'av' in rl\_actions.keys():  reward['av'] = rewards.min\_delay\_unscaled(self)  return reward |

代码1-2的主要功能是从环境中获取需要的排队长度、平均速度等指标来衡量当前网络状态的好坏，并通过对数值的处理来满足相应的激励与惩罚来促进交通信号灯和自动驾驶汽车协调与优化。

#### 1.2.2 强化学习算法的学习

在强化学习中的策略根据输入的状态决定智能体在时刻将采取什么样的动作 ， 假设 代表策略 的参数那么智能体的目标就是找到能够使得期望回报（expected return）最大的。

### 1.3 职业素养学习培养

（描述对软件工程系统的质量、环境、职业健康、安全和服务意识的学习和认识，对职业道德和规范的理解和遵守。）

### 1.4. 工程协作与交流情况

（阐述在实习执行过程中，针对特定的目标或问题，与工程项目组成员，包括与其他学科的成员合作并开展工作的情况。）

## 2. 存在问题与解决方案

### 2.1. 存在的主要问题

（分析、总结和归纳实习执行过程中存在的主要问题与具体的应对措施，能通过具体事例来说明在心理素质培养以及应对项目风险与挑战的能力锻炼等方面是否有所收获。）

### 2.2. 解决方案与可行性研究

（针对当前尚未解决的复杂工程问题，分析文献寻求解决方案，并在安全、环境、法律等约束条件下，通过技术经济评价对设计方案的可行性进行研究，说明方案的合理性。）

## 3. 前期任务完成度与后续实施计划

（前期任务完成度自我评价及后续工程环节的实施计划安排。除针对复杂工程问题的实施方案执行计划，还包括针对个人或职业发展需求的知识技能学习计划。）

## 参考文献

[1] 王浩刚，聂在平.三维矢量散射积分方程中奇异性分析[J]. 电子学报，1999, 27(12): 68-71

[2] X. F. Liu, B. Z. Wang, W. Shao. A marching-on-in-order scheme for exact attenuation constant extraction of lossy transmission lines[C]. China-Japan Joint Microwave Conference Proceedings, Chengdu, 2006, 527-529

[3] 竺可桢.物理学[M]. 北京：科学出版社，1973, 56-60