# 基于深度强化学习的交通信号控制

乔志敏[1], 柯良军[2†]

(1. 太原工业学院 自动化系, 山西 太原 030008;

2. 西安交通大学 自动化科学与工程学院, 机械制造系统工程国家重点实验室, 陕西 西安 710049)

**摘要**: 当前广泛应用的基于车流动力学建模的交通信号优化模型精确度较高, 但迁移能力稍弱, 针对该问题, 本文提出了一种基于深度强化学习的单智能体交通信号控制方法. 该方法首次在考虑交叉口有行人穿越干扰的情况下定义了动作空间, 从3个不同的角度定义了3种奖励函数, 并提出了一种累积延迟近似方法. 在算法方面, 提出了一种基于动态权重的Soft Actor-Critic算法, 该算法可以动态调整Actor网络和Critic 网络的更新幅度, 显著地提高了传统Soft Actor-Critic算法的收敛效率和收敛性能. 仿真结果表明, 本文提出的模型和算法在降低车辆延迟时间、减少车辆停车次数以及减少车辆队列长度等交通性能指标方面是有效的.

**关键词**: 交通信号; 强化学习; 动态权重; 延迟时间

**引用格式**: 乔志敏, 柯良军. 基于深度强化学习的交通信号控制. 控制理论与应用, 2025, 42(1): 76 – 86

DOI:10.7641/CTA.2023.30274

# Traffic signal control based on deep reinforcement learning

QIAO Zhi-min[1], KE Liang-jun[2†]

(1. Department of Automation, Taiyuan Institute of Technology, Taiyuan Shanxi 030008, China;
2. State Key Laboratory for Manufacturing System Engineering, School of Automation Science and Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China)

**Abstract:** The widely used traffic signal collaborative optimization model based on vehicle flow dynamics modeling has high accuracy but slightly weak transfer ability. To address this issue, this paper proposes a single agent traffic signal control method based on deep reinforcement learning. This method defines the action space for the first time considering pedestrian crossing interference at intersections, and defines three reward functions from three different perspectives, and proposes a cumulative delay approximation method. In terms of algorithm, a dynamic weight based soft actor-critic algorithm has been proposed, which can dynamically adjust the update amplitude of the actor network and critic network, significantly improving the convergence efficiency and performance of traditional soft actor-critic algorithm. The simulation results show that the proposed model and algorithm can effectively improve traffic performance indicators, such as reducing vehicle delay time, reducing vehicle parking times, and reducing vehicle queue length.

**Key words:** traffic signal; reinforcement learning; dynamic weight; delay time

**Citation:** QIAO Zhimin, KE Liangjun. Traffic signal control based on deep reinforcement learning. *Control Theory & Applications*, 2025, 42(1): 76 – 86

## 1 引言

随着车辆数量的急剧增长, 城市交通信号控制(traffic signal control, TSC)变得越来越复杂. 虽然通过新建道路和扩建原有道路可以在一定程度上缓解交通拥堵问题, 但由于土地资源是有限的, 不可能无限扩张以应对不断增大的车辆规模. 因此, 通过合理控制交通网内各交叉口信号灯配时和信号相序来适当地调节交通流量, 从而提高现有基础设施的利用率, 就成为一种减少交通拥堵至关重要的方法.

许多科研工作者致力于TSC的研究, 目的是最大限度地减少整个交通网中的车辆的平均等待时间和停车次数[1]. 传统的交通信号控制方法有Maxband[2],

收稿日期: 2023−04−26; 录用日期: 2023−11−22.

†通信作者. E-mail: keljxjtu@xjtu.edu.cn; Tel.: +86 18706796441.

本文责任编委: 高会军.

山西省教育厅高等学校科技创新项目(2022L523), 国家自然科学基金项目(61973244, 72001214), 山西省基础研究计划资助项目(202303021222300), 第五届太原工业学院学科带头人资助项目资助.

Supported by the Shanxi Province Department of Education University Science and Technology Innovation Project (2022L523), the National Natural Science Foundation of China (61973244, 72001214), the Fundamental Research Program of Shanxi Province (202303021222300) and the 5th Discipline Leader Project of Taiyuan Institute of Technology.

Multiband[3]等, 但这些都属于定时控制, 当交通网规模较大时, 该方法无法动态地调整信号配时, 也无法对交通流量进行预测. 在自适应交通信号控制方面, 配时、周期和相位差优化技术 (split, cycle and offset optimizmion technique, SCOOT) 和悉尼协调自适应交通系统 (sydney coordinated adaptive traffic system, SCATS) 等系统虽然可以动态地调整信号配时, 但车流信息的收集采用被动方式, 因此无法预测交通流的变化. PREDICT方法[4]通过在上游交叉口设置检测器, 虽然能够根据上游的信号相位和队列长度预测交通流量, 但是PREDICT方法仅在当上游交叉口为定时信号控制时才有效. 另外, Zhu等人[5]利用GPS数据开发了一种到达时间估计模型, 将公交到达时间作为后验估计参数. Qiao等人[6]通过采集相邻交叉口的实时交通流信息, 建立了一种基于半分布式的自适应交通信号配时和流量预测模型. Liu等人[7]建立一种基于交通信号相位差的车流延迟模型, 把空间信息细分为交叉口信息和交通流向序列信息, 融合时空特征来进行交通信号配时, 但这些自适应TSC方法建立的车流动力学模型通常较复杂, 对模型进行优化非常困难, 仍然存在许多的局限性.

深度强化学习作为解决交通信号控制问题的另一类方法, 通过学习控制行为和由此产生的车辆变化对动态的复杂交通系统进行隐式建模, 从学习到的输入–输出对中寻找最优信号配时方案. Genders等人[8]使用异步的优势演员–评论家算法研究了不同的状态表示对交叉口信号控制的影响, 并在动态交通仿真环境中分别对3种不同的状态定义做了测试. Genders等人[9]最近研究了交通信号控制的异步深度强化学习模型. Li等人[10]提出了一种基于自动编码器的深度强化学习算法, 用于解决具有动态交通流的单交叉口信号控制问题. Choe等人[11]提出了一种用于单交叉口信号控制的基于循环神经网络(recurrent neural network, RNN) 的深度 Q 网络(deep Q-network, DQN) 模型. 然而, 这些方法中还存在一些问题, 例如交通信号动作空间设计不合理、奖励函数设计不合理导致交通指标下降、算法设计不合理导致收敛效率低及收敛性能差等. 本文以接近信号交叉口的车辆以及单交叉口的信号灯的状态为观察对象, 首先, 对单交叉口信号控制问题的状态空间、动作空间以及奖励函数进行了重新设计, 更加充分地利用了交叉口附近车辆的信息, 从微观层面实现了交通信号控制, 改善了交通性能指标; 其次, 本文提出了一种全新的强化学习算法, 该算法首次应用于交通信号控制问题中, 并解决了目前主流的深度强化学习算法收敛效率低和收敛性能差的问题.

本文的主要贡献包括:

1) 首次在有行人穿越交叉口的情况下定义了动作空间. 行人穿越交叉口是不可避免的因素, 如果不能合理处理行人问题, 不仅威胁到行车安全, 更会显著地增大车辆延迟. 本文通过在绿灯相位中加入闪烁禁止步行时间, 避免了潜在的安全问题以及由此产生的延迟问题.

2) 从不同的角度刻画了3种奖励函数, 并通过后续仿真分析了3种奖励函数对交通性能指标以及相应算法的影响.

3) 提出了一种累积延迟近似方法, 使得接近交叉口的车辆累积延迟测量变得切实可行, 并通过仿真证明该方法是有效的.

4) 提出了一种基于动态权重的Soft Actor-Critic算法. 该算法中加入的动态权重机制在智能体采取的动作有助于系统性能的提高时, 则增强更新范围, 否则减小更新范围, 显著地提高了算法的收敛效率和收敛性能.

本文的组织结构如下: 第2节给出了基于马尔可夫决策过程的交通信号控制问题描述; 第3节给出基于动态权重的Soft Actor-Critic算法设计; 第4节是仿真与分析; 最后, 在第5节进行总结.

## 2　基于马尔可夫决策过程的交通信号控制问题描述

### 2.1　状态空间

车辆的队列长度能相对紧凑和全面地反映交叉口的交通情况. 然而, 仅仅把队列长度作为环境状态是不够的. 智能体必须知道当前所处的相位状态, 即当前绿灯相位, 以便做出合理的决策. 相位的切换对交叉口的状态影响很大, 如果智能体没有把当前绿灯相位作为环境状态的一部分, 它就不可能知道哪个动作延长了当前相位时间, 哪个动作切换到了其他相位. 因此, 除了队列长度之外, 还应在环境状态空间中加入当前绿灯相位状态$P_{\mathrm{g}}^{k}$.

另外, 帮助智能体找到最优动作策略的最后一个信息是当前的绿灯信号已经经过的时间$E_{\mathrm{g}}^{k}$. 当某一相位的信号灯切换到绿灯时, 该相位方向上的车队开始移动并逐渐消散. 此时, 如要判断车队是否已经全部通过交叉口就需要用到时间$E_{\mathrm{g}}^{k}$. 综合以上情况将状态空间定义为

$$S^{k}=\left\{q_{1}^{k},\cdots,q_{m}^{k},P_{\mathrm{g}}^{k},E_{\mathrm{g}}^{k}\right\},\tag{1}$$

式中: $q_{m}^{k}$是时间步为$k$, 车辆行驶方向为$m$时的车辆数; $P_{\mathrm{g}}^{k}$是当前绿灯相位; $E_{\mathrm{g}}^{k}$是当前绿灯相位经过的时间.

### 2.2　动作空间

本文选择基于可变相序的交通信号切换策略作为动作空间, 相位动作选择如表1所示.

这里可以将智能体的动作分为两种, 一种是延长

当前相位的绿灯时长; 另一种是切换到下一个相位. 如果当前绿灯相位是相位$i \in \Phi$($\Phi$是可能相位的集合), 并且智能体选择的下一个相位仍然是$i$时, 则把当前绿色相位$i$的时长延长1 s. 如果当前的绿色相位是$i$, 并且智能体将选择另一个绿灯相位$j \neq i, \forall j \in \Phi$, 则该信号必须经过预定义的黄灯时间$y$、全红灯时间$R$和切换到下一个绿灯相位前的最短绿灯时间$G_j$, 在此期间, 智能体不采取任何动作.

表 1 相位动作选择

Table 1 Phase action selection

| 相位序号 | 车辆行驶方向 | 相位示意 | 相位序号 | 车辆行驶方向 | 相位示意 |
|---|---|---|---|---|---|
| 1 | NL + SL |  | 5 | EL + WL |  |
| 2 | S + SL |  | 6 | W + WL |  |
| 3 | N + NL |  | 7 | E + EL |  |
| 4 | S + N+ SL + NL |  | 8 | W + E+ WL + EL |  |

在这种情况下, 动作空间为

$$A = \{1, 2, \cdots, P\}, \tag{2}$$

式中$P$是可供选择的相位序号.

间隔时间可以表示为

$$\Delta t = \begin{cases} y + R + G_j, & a^k = j \neq a^{k-1}, \\ 1, & a^k = a^{(k-1)}, \end{cases} \tag{3}$$

式中$a^k$是智能体在时间步$k$采取的动作. $G_j$通常是根据行人穿越交叉口的安全时间确定的. 行人安全时间由两部分组成: 步行时间间隔和闪烁禁止步行时间间隔. 步行时间间隔是行人收到的可正常步行的信号时间, 通常至少持续4 s. 闪烁禁止步行时间是为了让行人有足够的时间在信号改变之前离开交叉口并到达街道的另一侧, 其保证行人能够安全地通过交叉口, 计算公式如下:

$$\text{fdw} = \frac{w}{s_w}, \tag{4}$$

式中: $w$是街道的宽度; $s_{\text{w}}$是平均步行速度(通常选择1 m/s).

### 2.3 奖励函数

本文定义了3种不同的奖励函数. 交通信号控制最重要的指标之一是车辆通过交叉口之前排队等候的时间, 即车辆延迟时间, 本文定义3种奖励函数的目的就是通过仿真分析选择最优的一种来最大限度地减少交叉口的车辆延迟时间.

1) 奖励函数1.

奖励函数1(Reward 1, R1)定义为交叉口处两个连续动作之间累积延迟差值的平均变化率. 每当车辆接近交叉口时, 该车辆在环境中被监控以记录其速度和延迟. 因此, 在每个时间步, 都有一个接近该交叉口的所有车辆的清单$\text{VL}^t = \{u \mid$时间步为$t$时, 在交叉口的车辆$u\}$记录其速度$v_u^t$和延迟$d_u^t$. 根据车辆的行驶方向来区分交叉口的车辆, 并用$M$来表示车辆在交叉口的行驶轨迹, 在一个标准的四方向交叉口, $M = \{\text{N}, \text{NL}, \text{S}, \text{SL}, \text{W}, \text{WL}, \text{E}, \text{EL}\}$. 其中: N, S, W, E分别代表北行、南行、西行、东行, L代表左转, 从而有$\text{VL}^t = \bigcup\limits_{m \in M} \text{VL}_m^t$. 因此, 交叉口在时间步$t$的累积延迟$\text{CD}^t$可以表示为

$$\text{CD}^t = \sum_{u \in \text{VL}^t} d_u^t = \sum_{m \in M} \sum_{u \in \text{VL}_m^t} d_u^t = \sum_{m \in M} \text{CD}_m^t, \tag{5}$$

式中$\text{CD}_m^t$表示时间步为$t$, 行驶方向为$m$时的车辆累积延迟.

接下来, 计算每辆车的延迟$d_u^t$. 这里只考虑车辆在排队时被延迟, 即因为交通信号而被延迟. 因此, 本文引入变量$\text{inq}_u^t$来表示在时间步为$t$时车辆是否在队列中. 只有当车辆速度$\text{sp}_u^t$低于预先定义的队列速度阈值$\text{sp}_{\text{q}}$时, 车辆才被视为在队列中. 变量$\text{inq}_u^t$可以表示为

$$\text{inq}_u^t = \begin{cases} 1, & \text{sp}_u^t < \text{sp}_{\text{q}}, \\ 0, & \text{sp}_u^t \geqslant \text{sp}_{\text{q}}. \end{cases} \tag{6}$$

因此有

$$d_u^t = d_u^{t-1} + \text{inq}_u^t, \ d_u^0 = 0, \ \forall u \in \text{VL}^t. \tag{7}$$

很明显, 当车辆通过停车线并离开交叉口时, 它将不再位于交叉口的车辆集合中$\text{VL}^t$中. 在每个时间步$t$交叉口都有累积延迟, 因此奖励函数可以表示为

$$r^k = \begin{cases} \text{CD}^{k-1} - \text{CD}^k, & a^k = a^{k-1}, \\ \dfrac{\text{CD}^{k-1} - \text{CD}^k}{y + R + G_j}, & a^k = j \neq a^{k-1}. \end{cases} \tag{8}$$

2) 奖励函数2.

奖励函数2(Reward 2, R2)定义为交叉口队列长度的总和. 首先来关注车辆在交叉口的某一次行驶轨迹, 为了简单起见, 假设与交叉口相连接的道路为单车道. 如图1所示为与交叉口相连的一条道路在不同时间步的车辆队列情况. 蓝色的车辆在队列中处于停止状态, 橙色的车辆处于移动状态, 红色虚线表示静止车辆队列的后部, 黑色虚线表示静止车辆队列的前部. 两条线之间的距离就是车辆队列长度.

图1显示了车辆如何开始排队以及上游车辆如何加入队列. 当信号灯变为绿色时, 排在最前面的车辆开始移动, 队列开始缩小. 在图1中有两条假想的线, 在不同的时间步标记队列的头部$Q_{\text{f}}$和尾部$Q_{\text{e}}$. 车辆队列的末端从红色信号开始, 随着车辆加入队列而向

上移动. 上游的车流量决定了红色虚线的斜率, 上游车流量越大, 排队的速度就越快, 斜率越大. 类似地, 队列的头部从绿色信号灯开始, 随着车辆的加速移动, 不再被认为在队列中. 黑色虚线的斜率由交叉口的出口流量决定, 通常比红色虚线的末端更陡. 否则, 队列不会消失, 而是不断增长. 从该图可以看出, 1号车辆的延迟是其在队列中花费的时间, 即$t_5-t_1$或者是$Q_{\mathrm{f}}$和$Q_{\mathrm{e}}$之间的水平距离, 其他车辆的延迟同理.

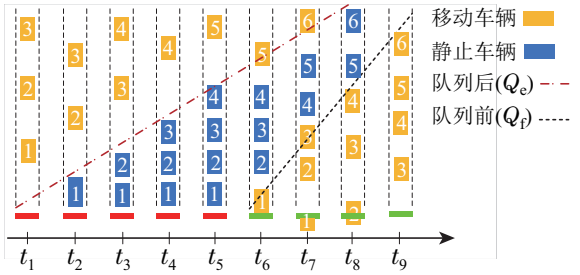



图 1　与交叉口相连的一条道路在不同时间步的车辆队列情况

Fig. 1　The queue of vehicles at different time steps on a road connected to an intersection

图2为与交叉口相连的道路的累积延迟示意图. 道路的累计延迟是该车道上车辆延迟的总和, 也可以通过对队列长度求和来计算延迟. 周期1中的车队移动累积延迟$\mathrm{CD}_m^{\mathrm{C1}}$是该周期内所有排队车辆的延迟的总和$\mathrm{CD}_m^{\mathrm{C1}}=d_1+d_2+d_3+d_4+d_5+d_6$. 该延迟总和等于队列的头部$Q_{\mathrm{f}}$和尾部$Q_{\mathrm{e}}$两条线之间的面积$A_m^1$. 类似地, 对任意的信号周期$k$, 在该行驶轨迹下, 排队车辆产生的累积延迟都等于这两条线以及时间轴围成区域的面积. 因此, 本文的目标就是最小化这些区域的总和, 从而最小化该交叉口的累积延迟.

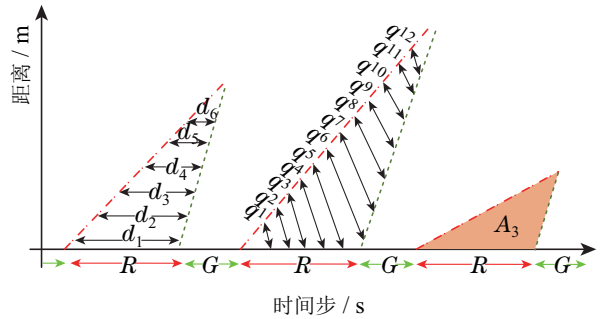



图 2　与交叉口相连的道路的累积延迟

Fig. 2　Accumulated delay of roads connected to intersections

在图2中, 对于第2个公共周期, 将图分成不同的时间步, 对于每个时间步, 队列的前部和尾部之间的距离显然是队列长度. 如果将不同时间步的所有队列长度加在一起, 会得到$Q_{\mathrm{f}}$和$Q_{\mathrm{e}}$之间的面积, 可以表示为

$$\mathrm{CD}=\sum_{m\in M}\mathrm{CD}_m=\sum_{m\in M}\sum_{n=1}^{C}A_m^n=\sum_{m\in M}\sum_{t=1}^{T}q_m^t, \tag{9}$$

$$q_m^t=\sum_{u\in \mathrm{VL}_m^t}\mathrm{inq}_u^t, \tag{10}$$

式中: CD是交叉口的累积延迟; $C$是一个仿真时段的周期数; $T$是一个仿真时段的时间步数. 由式(9)可知, 为了最小化交叉口的累积延迟, 可以简单地将所有移动的车队长度之和最小化. 因此, 在任意时间步下所有方向的车辆队列长度的总和可以定义为奖励函数, 用于最小化交叉口的累积延迟, 可以定义为

$$r^k=\begin{cases}-1\times\sum\limits_{m\in M}q_m^t, & a^k=a^{k-1},\\ -1\times\sum\limits_{m\in M}\sum\limits_{p=t}^{t+y+R+G_j}q_m^p, & a^k=j\neq a^{k-1}.\end{cases} \tag{11}$$

3) 奖励函数3.

奖励函数3 (Reward 3, R3) 定义为交叉口累积延迟的差, 这与第1个奖励函数非常相似, 从奖励函数1可以看到, 如果智能体采取的动作是当前相位信号的扩展, 奖励仅仅是交叉口累积延迟的差值. 如果智能体采取的动作是切换信号相位, 则奖励是累积延迟的差值除以下一相位的黄灯、全红灯和最小绿灯时间的和. 这种区分方式会导致一个无法避免的问题, 即智能体采取扩展动作与切换动作的奖励具有不同的量纲, 扩展动作的量纲是秒, 切换动作的量纲是秒/秒(即没有单位). 如果将这两种具有不同量纲的奖励相互叠加是不合逻辑的.

此外, 假设智能体选取的动作是切换到另外一个相位. 这种情况下包括两个过程: 第一是智能体在较长时间 $y+R+G_j$下处于保持状态, 期间当信号灯为红色时车辆排队导致的累积延迟显著增加; 另一个是当信号灯变为绿色时, 队列最前面的车辆(对累积延迟贡献最大)已经开始移动, 并且很可能已经通过了交叉口, 此期间累积延迟显著减少. 这两个过程是发生在交叉口的车辆延迟最关键的两个过程. 因此, 智能体应该充分感受到这两种过程的影响(实际为绿信比的影响). 用两个周期内累积奖励的差除以智能体保持期(无行动期)会削弱这两个过程的重要性. 因此, 本文提出第3个奖励函数, 其可以表示为

$$r^k=\mathrm{CD}^{k-1}-\mathrm{CD}^k,\ \forall a^k\in A, \tag{12}$$

式中$A$是动作空间.

## 2.4　累积延迟的近似

在R1和R3中, 需要跟踪与交叉口相连的道路上所有接近交叉口的车辆. 每辆车的延迟, 即它在队列中花费的时间都要被储存. 然而, 在实际交通环境中这种做法有点不切实际, 由于庞大的数据量和计算量, 利用现有技术不可能实现实时存储并调取每辆车的延迟信息. 因此, 在不考虑每辆车实际延迟的情况下, 提出了一种可以近似奖励函数1和奖励函数3中车辆延迟的方法. 该方法仅仅需要队列长度$q_m^t$和交叉口的输出车流$O_m^t$即可. 本文引入了一个全新的辅助变量

$z_m^t, m\in M$, 用于表示导致延迟的车辆, 其可以表示为

$$z_m^t=\begin{cases} q_m^t, & \text{信号灯为红色},\\ z_m^{t-1}-O_m^t, & \text{信号灯为绿色}, \end{cases} \tag{13}$$

式中: $m$是车辆的行驶方向; $t$是时间步.

在该方法中, 当交通灯是红色时, 根据队列中的车辆数量来计算延迟. 当信号灯变为绿色时, 只关注那些在红灯期间进入队列的车辆, 并假设每辆车产生的延迟是均匀分布的. 如果在当前行驶方向上有$O_m^t$辆离开了交叉口, 则意味着仍然有$z_m^{t-1}-O_m^t$辆车在红灯期间被延迟. 假设交叉口车队产生的延迟与停留在交叉口的车辆数是成正比的, 那么, 当有$O_m^t$辆车离开交叉口时, 延迟时间$\mathrm{CD}_m^t$就减少$\dfrac{O_m^t}{z_m^{t-1}}$, 则延迟时间$\mathrm{CD}_m^t$可以表示为

$$\widehat{\mathrm{CD}_m^t}=\begin{cases} \widehat{\mathrm{CD}_m^{t-1}}+q_m^t, & \text{信号灯为红色},\\ (1-\dfrac{O_m^t}{z_m^{t-1}})\mathrm{CD}_m^{t-1}, & \text{信号灯为绿色}. \end{cases} \tag{14}$$

这种近似方法解决了奖励函数难以观察和计算的问题. 在后续仿真中, 本文将进一步评估该近似方法对智能体性能的影响.

## 3 基于动态权重的Soft Actor-Critic算法

Soft Actor-Critic(SAC)算法在解决许多实际问题中都显示出了良好的效果, 并得到了广泛的应用. SAC算法是面向最大熵 (maximum entropy, ME) 强化学习开发的一种离线策略算法, 和深度确定性策略梯度算法 (deep deterministic policy gradient, DDPG) 相比, SAC使用的是随机策略, 相比确定性策略具有一定的优势. 具体来说, 确定性策略是指这个策略对于一种状态只考虑一个最优的动作, 而在许多问题中, 最优的动作可能不止一个, 此时就可以考虑给出一个随机策略, 在每一个状态上都能输出每一种动作的概率, 比如有3个动作都是最优的, 概率一样都最大, 那么就可以从这些动作中随机选择一个做为输出. 而最大熵的核心思想就是不遗落任意一个有用的动作. DDPG采用确定性策略的做法是看到一个好的就捡起来, 差一点的就不要了, 而最大熵是都要捡起来, 都要考虑.

然而SAC采用的是固定学习速率, 智能体无法根据即时奖励随时间步的变化来动态调整学习率, 这在一定程度上影响了算法的收敛速率. 为此, 本文将动态权重引入SAC算法中, 提出了一种新的深度强化学习算法叫做基于动态权重的Soft Actor-Critic算法(dynamic weights SAC, DWSAC). 当智能体采取的动作明显有助于系统性能的提高时增强更新范围, 否则削弱更新范围, 显著提高了算法的收敛效率和收敛性能.

### 3.1 动态权重

为了在SAC算法中引入动态权重, 首先, 需要区分在学习过程中的有用和无用信息. 根据文献[12–13]发现, 在Actor-Critic框架中, 智能体收集的大部分信息对Critic来说都没有价值. 原因在于, 强化学习中, 智能体获得的信息是稀疏的、延迟的, 这导致在大多数情况下, 智能体不能有效地获得有用的奖励值. 此外, 如何让Actor从稀疏的奖励值中学习动作决策, 是另一个需要解决的重要问题, 因此, 需要修改算法的参数更新过程. 为了更准确地更新参数, 需要为智能体的用于更新网络参数的梯度分配一个权重. 由于Actor和Critic参数更新的特点不同, 因此, 需要分别为Actor和Critic分配不同的更新权重, 接下来, 将详细介绍生成权重的过程.

首先, 来介绍Critic网络的权重, 根据智能体动作执行前后的奖励值, 为网络参数的更新设置一个比值, 用于反映当前动作对环境影响的大小, 提高算法的收敛速率. 需要注意的是, 在该比值的定义过程中, 可以简单地使用当前奖励$R_{\mathrm{cur}}$与先前奖励$R_{\mathrm{prv}}$的比, 但是当它们相似时, 仅使用线性比不能很好地反映更新的值. 因此, 将比值定义为

$$\mathrm{Ratio_c}=\begin{cases} 1, & R_{\mathrm{prv}}=0,\\ \exp(\dfrac{R_{\mathrm{cur}}}{R_{\mathrm{prv}}}-1), & R_{\mathrm{cur}}/R_{\mathrm{prv}}>1,\\ R_{\mathrm{cur}}/R_{\mathrm{prv}}, & R_{\mathrm{cur}}/R_{\mathrm{prv}}\leqslant 1. \end{cases} \tag{15}$$

有了比值的定义, 就可以用它来给梯度赋权重. 根据文献[14]中的Critic网络参数更新公式$\theta_i\leftarrow\theta_i-\lambda_Q\hat{\nabla}_{\theta_i}J_Q(\theta_i), i\in\{1,2\}$和式(15), Critic网络参数的更新可以改写为如下形式:

$$\theta_i\leftarrow\theta_i-\varepsilon_{\mathrm{c}}\cdot\min(\mathrm{Ratio_c},\xi_{\mathrm{c}})\cdot\hat{\nabla}_{\theta_i}J_Q(\theta_i),\ i\in\{1,2\}, \tag{16}$$

式中: $\theta_i$是Critic网络的参数; $\varepsilon_{\mathrm{c}}$是Critic网络的学习率. 为了防止在一次更新有较大的变化, 还提出了一个阈值$\xi_{\mathrm{c}}$来表示梯度更新权重的上限.

至于Actor网络的权重, 研究中发现如果当前奖励$R_{\mathrm{cur}}$与先前奖励$R_{\mathrm{prv}}$的变化量很大, 那么Actor需要大幅度更新, 但是如果更新幅度太大, 神经网络就会产生振荡. 因此, 定义了一种平滑的方法来计算Actor的梯度权重, 其可以定义为

$$\mathrm{Ratio_a}=1+\frac{|R_{\mathrm{cur}}-R_{\mathrm{prv}}|^2}{R_{\mathrm{cur}}^2+R_{\mathrm{prv}}^2}. \tag{17}$$

根据文献[14]中的Actor参数更新公式$\phi\leftarrow\phi-\lambda_\pi\hat{\nabla}_\phi J_\pi(\phi)$和式(17), Actor的网络参数更新可以改写为如下形式:

$$\phi\leftarrow\phi-\varepsilon_{\mathrm{a}}\cdot\min(\mathrm{Ratio_a},\xi_{\mathrm{a}})\cdot\hat{\nabla}_\phi J_\pi(\phi), \tag{18}$$

这里仍然使用阈值$\xi_{\mathrm{a}}$来避免Actor网络中的震荡. 采用这种方式之后, 可以高效地利用有效的资源来加快网络的收敛.

### 3.2 基于动态权重的Soft Actor-Critic算法设计

SAC算法使用函数逼近器对软$Q$值和策略进行逼近, 并使用随机梯度下降来优化两个网络. 参数化之后的$Q$值函数和策略函数分别为$Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)$ 和$\pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, 其网络参数分别是$\theta$和$\phi$. 接下来将为这些参数导出更新规则.

软状态值函数$V(\boldsymbol{s}_t)$可以定义为

$$V(\boldsymbol{s}_t) = \mathrm{E}_{\boldsymbol{a}_t\sim\pi}[Q(\boldsymbol{s}_t, \boldsymbol{a}_t) - \alpha\log\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)], \quad (19)$$

软$Q$值函数的参数可以通过最小化软贝尔曼残差来训练, 其可以表示为

$$J_Q(\theta) = \mathrm{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\mathcal{D}}[\frac{1}{2}(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - (r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma\mathrm{E}_{\boldsymbol{s}_{t+1}\sim p}[V_{\bar{\theta}}(\boldsymbol{s}_{t+1})]))^2], \quad (20)$$

式(20)中的价值函数$V_{\bar{\theta}}(\boldsymbol{s}_{t+1})$是通过式(19)的软$Q$值函数的参数$\theta$隐式参数化后的形式, 用随机梯度对式它进行优化, 可以表示为

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\boldsymbol{a}_t, \boldsymbol{s}_t)(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - (r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma(Q_{\bar{\theta}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) - \alpha\log(\pi_\phi(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1}))))), \quad (21)$$

该更新利用了具有参数$\bar{\theta}$的目标软$Q$值函数.

对于策略, SAC利用库尔巴克–莱布勒散度对其进行更新, 可以表示为

$$\pi_{\text{new}} = \arg\min_{\pi'\in\Pi} D_{\text{KL}}(\pi'(\cdot|\boldsymbol{s}_t)\| \frac{\exp(\frac{1}{\alpha}Q^{\pi_{\text{old}}}(\boldsymbol{s}_t, \cdot))}{Z^{\pi_{\text{old}}}(\boldsymbol{s}_t)}), \quad (22)$$

式中$Z^{\pi_{\text{old}}}(s_t)$是用于归一化分布的配分函数, 其通常很难处理, 但它对新策略的梯度没有贡献, 通常可以忽略.

策略$\pi$的参数$\theta$可以通过直接最小化式(22)中的期望库尔巴克–莱布勒散度来学习, 可以表示为

$$J_\pi(\phi) = \mathrm{E}_{\boldsymbol{s}_t\sim\mathcal{D}}[\mathrm{E}_{\boldsymbol{a}_t\sim\pi_\phi}[\alpha\log(\pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)) - Q_\theta(\boldsymbol{a}_t, \boldsymbol{s}_t)]], \quad (23)$$

式中$\alpha$是一个常数, 其决定了熵项相对于奖励的相对重要性.

最小化$J_\pi$有多种方法, 策略梯度法的一种典型解决方案是使用似然比梯度估计[15], 该方案不需要通过策略和目标密度网络反向传递梯度. 然而, 在本文中, 目标密度是由神经网络表示的$Q$函数并且可以被微分. 因此, 改为使用再参数化方法, 这不仅方便而且方差估计更低. 为此, 利用神经网络重新参数化了策略, 可以表示为

$$\boldsymbol{a}_t = f_\phi(\epsilon_t, \boldsymbol{s}_t), \quad (24)$$

式中$\epsilon_t$是输入噪声, 从某个固定分布(如球形高斯分布)中采样.

根据式(24), 式(23)可以改写为如下形式:

$$J_\pi(\phi) = \mathrm{E}_{\boldsymbol{s}_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}[\alpha\log\pi_\phi(f_\phi(\epsilon_t, \boldsymbol{s}_t)|\boldsymbol{s}_t) - Q_\theta(\boldsymbol{s}_t, f_\phi(\epsilon_t, \boldsymbol{s}_t))], \quad (25)$$

式中$\pi_\phi$是根据$f_\phi$隐式定义的.

式(24)的梯度可以表示为

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi\alpha\log(\pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)) + (\nabla_{\boldsymbol{a}_t}\alpha\log(\pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)) - \nabla_{\boldsymbol{a}_t}Q(\boldsymbol{s}_t, \boldsymbol{a}_t))\nabla_\phi f_\phi(\epsilon_t, \boldsymbol{s}_t), \quad (26)$$

式中$\boldsymbol{a}_t$在$f_\phi(\epsilon_t; \boldsymbol{s}_t)$中被评估. 这种无偏的梯度估计将DDPG形式的策略梯度[16]扩展到任何易处理的随机策略.

前述算法是在给定温度的前提下学习最大熵策略, 但是在实际问题中最佳温度应该根据具体问题来调整. 因此, 制定一个最大熵强化学习目标, 自适应地调整温度具有实际意义, 其中熵被视为一个约束, 在该约束中策略的平均熵受到约束, 而不同状态下的熵是不同的. 算法的目标是找到一个奖励期望最大的随机策略, 并且, 该策略满足熵约束的期望最小, 可以表示为

$$\max_{\pi_{0:T}} \mathrm{E}_{\rho_\pi}[\sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t)],$$
$$\text{s.t. } \mathrm{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\rho_\pi}[-\log(\pi_t(\boldsymbol{a}_t|\boldsymbol{s}_t))]\geqslant\mathcal{H},\ \forall t, \quad (27)$$

式中$\mathcal{H}$是熵的期望的最小值. 需要注意的是, 对于可完全观测的马尔可夫决策过程, 优化奖励期望的策略是确定性的, 因此, 该约束通常是严格的, 不需要对熵施加上限.

由于在时间步$t$的策略只能影响未来的目标值, 因此, 可以采用动态规划方法. 这里将目标改写为迭代最大化的形式

$$\max_{\pi_0}(\mathrm{E}[r(\boldsymbol{s}_0, \boldsymbol{a}_0)] + \max_{\pi_1}(\mathrm{E}[\cdots] + \max_{\pi_T}\mathrm{E}[r(\boldsymbol{s}_T, \boldsymbol{a}_T)])), \quad (28)$$

式(28)会受到熵的约束, 从最后一个时间步开始, 将约束最大化问题转化为对偶问题, 当

$$\mathrm{E}_{(\boldsymbol{s}_T,\boldsymbol{a}_T)\sim\rho_\pi}[-\log(\pi_T(\boldsymbol{s}_T|\boldsymbol{s}_T))]\geqslant\mathcal{H}$$

时, 其可以表示为

$$\max_{\pi_T}\mathrm{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\rho_\pi}[r(\boldsymbol{s}_T, \boldsymbol{a}_T)] = \min_{\alpha_T\geqslant 0}\max_{\pi_T}\mathrm{E}[r(\boldsymbol{s}_T, \boldsymbol{a}_T) - \alpha_T\log\pi(\boldsymbol{a}_T|\boldsymbol{s}_T)] - \alpha_T\mathcal{H}, \quad (29)$$

式中$\alpha_T$是对偶变量. 由于目标是线性的且约束(熵)在$\pi_T$中是凸函数, 所以, 这里还使用了强对偶, 该对偶目标与关于策略的最大熵目标密切相关, 最优策略是对

应于温度$\alpha_T$：$\pi_T^*(\boldsymbol{a}_T|\boldsymbol{s}_T;\alpha_T)$的最大熵策略. 最优对偶变量$\alpha_T^*$的求解可以表示为

$$\arg\min_{\alpha_T}\mathrm{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t\sim\pi_t^*}[-\alpha_T\log\pi_T^*(\boldsymbol{a}_T|\boldsymbol{s}_T;\alpha_T)-\alpha_T\bar{\mathcal{H}}], \tag{30}$$

为了简化, 利用了软$Q$值函数的递归定义

$$\begin{aligned}&Q_t^*(\boldsymbol{s}_t,\boldsymbol{a}_t;\pi_{t+1:T}^*,\alpha_{t+1:T}^*)=\\&\mathrm{E}[r(\boldsymbol{s}_t,\boldsymbol{a}_t)]+\mathrm{E}_{\rho_\pi}[Q_{t+1}^*(\boldsymbol{s}_{t+1},\boldsymbol{a}_{t+1})-\\&\alpha_{t+1}^*\log\pi_{t+1}^*(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1})],\end{aligned} \tag{31}$$

式中$Q_T^*(\boldsymbol{s}_T,\boldsymbol{a}_T)=\mathrm{E}[r(\boldsymbol{s}_T,\boldsymbol{a}_T)]$. 在熵的约束下再次使用对偶问题, 可以得到

$$\begin{aligned}&\max_{\pi_{T-1}}(\mathrm{E}[r(\boldsymbol{s}_{T-1},\boldsymbol{a}_{T-1})]+\max_{\pi_T}\mathrm{E}[r(\boldsymbol{s}_T,\boldsymbol{a}_T)])=\\&\max_{\pi_{T-1}}(Q_{T-1}^*(\boldsymbol{s}_{T-1},\boldsymbol{a}_{T-1})-\alpha_T\bar{\mathcal{H}})=\\&\min_{\alpha_{T-1}\geqslant 0}\max_{\pi_{T-1}}(\mathrm{E}[Q_{T-1}^*(\boldsymbol{s}_{T-1},\boldsymbol{a}_{T-1})]-\\&\mathrm{E}[\alpha_{T-1}\log\pi(\boldsymbol{a}_{T-1}|\boldsymbol{s}_{T-1})]-\alpha_{T-1}\bar{\mathcal{H}})+\alpha_T^*\bar{\mathcal{H}},\end{aligned} \tag{32}$$

这样就可以在时间上回溯, 递归地优化式(29). 需要注意的是, 在时间步$t$的最优策略是对偶变量$\alpha_t$的函数. 类似地, 可以在求解$Q_t^*$和$\pi_t^*$之后, 再求解最优对偶变量$\alpha_t^*$, 可以表示为

$$\begin{aligned}&\alpha_t^*=\\&\arg\min_{\alpha_t}\mathrm{E}_{\boldsymbol{a}_t\sim\pi_t^*}[-\alpha_t\log\pi_t^*(\boldsymbol{a}_t|\boldsymbol{s}_t;\alpha_t)-\alpha_t\bar{\mathcal{H}}].\end{aligned} \tag{33}$$

式(33)中的解以及前面描述的策略和软$Q$函数的更新构成了该算法的核心. 理论上, 准确地递归求解它们, 优化了式(29)中最优熵约束的最大期望奖励目标, 但实际上需要借助函数逼近器和随机梯度下降.

在本算法中, 利用两个软$Q$值函数来减轻策略改进步骤中的正偏差, 这种偏差会降低基于值的方法的性能, 具体来说是用参数$\theta_i$来参数化两个软$Q$值函数, 并且独立地训练它们来优化$J_Q(\theta_i)$, 接着把软$Q$值函数的最小值代入式(21)和式(26)中分别求解随机梯度和策略梯度.

除了软$Q$值函数和策略, 还通过最小化式(33)中的对偶目标来学习$\alpha$, 这可以通过近似双梯度下降来实现[17]. 尽管完全优化原始变量是不切实际的, 但在凸性假设下, 执行不完全优化的截断版本(甚至对于单个梯度步长)可以被证明是收敛的. 虽然这些假设不适用于神经网络等非线性函数逼近器的情况, 但在实践中发现这种方法仍然有效. 因此, 关于计算$\alpha$的梯度目标可以表示为

$$J(\alpha)=\mathrm{E}_{\boldsymbol{a}_t\sim\pi_t}[-\alpha\log\pi_t(\boldsymbol{a}_t|\boldsymbol{s}_t)-\alpha\bar{\mathcal{H}}]. \tag{34}$$

结合第3.1节中提出的动态权重, 本文提出了DWSAC算法, 算法的伪代码如算法1(见表2)所示.

表 2 算法1: DWSAC算法

Table 2 Algorithm 1: DWSAC algorithmde

**Input**: 初始参数$\theta_1,\theta_2,\phi$
**Output**: 训练后的参数$\theta_1,\theta_2,\phi$
初始化参数$\theta_1,\theta_2,\varphi$
初始化目标网络参数$\bar{\theta}_1\leftarrow\theta_1,\theta_2\leftarrow\bar{\theta}_2$
初始化一个空的经验回放池$\mathcal{D}\leftarrow\varnothing$
**for** 每一次迭代 **do**
  **for** 每个环境步 **do**
    根据当前策略获取动作$\boldsymbol{a}_t\sim\pi_\phi(\boldsymbol{a}_t\mid\boldsymbol{s}_t)$
    从环境中获得下一个状态$\boldsymbol{s}_{t+1}\sim p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)$
    储存到经验回放池$\mathcal{D}\leftarrow\mathcal{D}\cup\{\boldsymbol{s}_t,\boldsymbol{a}_t,r(\boldsymbol{s}_t,\boldsymbol{a}_t),\boldsymbol{s}_{t+1}\}$
  **end**
  **for** 每个梯度更新步 **do**
    更新$Q$值函数的参数$\theta_i\leftarrow\theta_i-\varepsilon_{\mathrm{c}}\times\min(\mathrm{Ratio}_{\mathrm{c}},\xi_{\mathrm{c}})\times\hat{\nabla}_{\theta i}J_Q(\theta_i)$, 其中$i\in\{1,2\}$
    更新策略参数$\phi\leftarrow\phi-\varepsilon_{\mathrm{a}}\times\min(\mathrm{Ratio}_{\mathrm{a}},\xi_{\mathrm{a}})\hat{\nabla}_\phi J_\pi(\phi)$
    调整温度$\alpha\leftarrow\alpha-\lambda\hat{\nabla}_\alpha J(\alpha)$
    更新目标网络参数$\bar{\theta}_i\leftarrow\tau\theta_i+(1-\tau)\bar{\theta}_i$, for $i\in\{1,2\}$
  **end**
**end**

## 4 仿真与分析

### 4.1 算法参数设置

本文选择了3种主流的深度强化学习算法: SAC, DDPG, 双延迟深度确定性策略梯度(twin delayed deep deterministic policy gradient, TD3), 与本文提出的DWSAC算法进行比较. DDPG的超参数设置与文献[16]保持一致, TD3的超参数设置与文献[17]保持一致, SAC的超参数设置与文献[14]保持一致, 本文提出的DWSAC算法的超参数设置如表3所示.

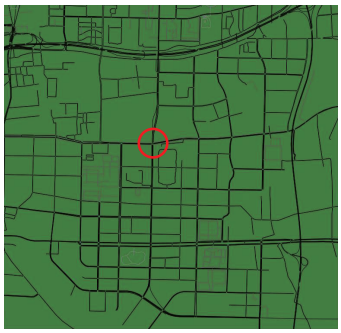
### 4.2 交通环境下的仿真与分析

如图3(a)所示, 在交通网中用红色圆圈标出了选定的单交叉口, 放大后如图3(b)所示, 该交叉口连接4条道路, 每条道路各包括两条车道. 为了评估算法在不同交通环境下的性能, 本文选取10个随机种子作为不同的交通环境分别测试所提出的算法及其比较算法的性能. 这里每个种子都代表现实生活中一天, 同一个交叉口每天的交通状况可能相似但不可能完全相同, 因此选取10个随机种子是合理的. 随后, 对10次仿真运行量进行平均, 用于评估算法的性能. 这里用于评估的性能指标除了奖励平均值、收敛速度以及鲁棒性之外, 还包括交叉口车辆的平均行驶时间、平均停车次数、在队列中的平均时间、平均队列长度以及队列长度标准差等.

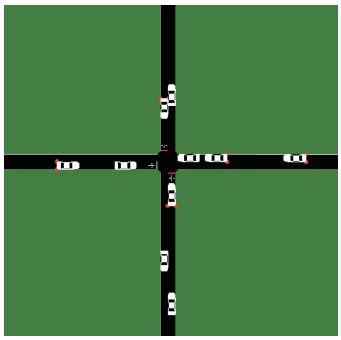
表 3 DWSAC算法的超参数设置

Table 3 Hyperparameter setting of DWSAC algorithm

| 超参数 | 值 |
|---|---|
| 优化器 | Adam |
| 值函数参数优化学习率 | 0.01 |
| 策略函数参数优化学习率 | 0.001 |
| 折扣率($\gamma$) | 0.99 |
| 经验缓存大小 | $1e+6$ |
| 每小批取样数 | 256 |
| 熵目标 | $-\dim(\mathcal{A})$ |
| 激活函数 | ReLU |
| 目标平滑系数($\tau$) | 0.005 |
| 目标更新间隔 | 1 |
| 梯度步 | 1 |
| Actor更新权重的阈值 | 10 |
| Critic更新权重的阈值 | 10 |
| 目标网络更新频率 | 300 |



(a) 红色圆圈标出选定的单交叉口



(b) 放大后的单交叉口

图 3 交通网环境仿真

Fig. 3 Traffic network environment simulation

1) 不同奖励函数对算法学习率、鲁棒性以及平均奖励的影响.

在3个不同的奖励函数下分别进行交通仿真, 奖励函数1、奖励函数2和奖励函数3对应的平均奖励曲线分别如图4–6所示.

从图 4–6 中的平均奖励曲线可以看出, 所提出的DWSAC算法在学习速率、鲁棒性以及平均奖励方面都要好于用于对比的3种主流算法.

2) 不同的强化学习算法对交通性能指标的影响.

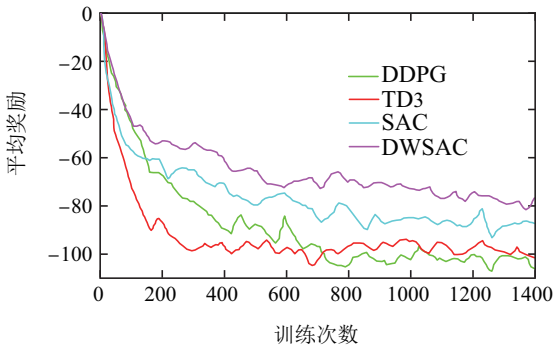



图 4 奖励函数为R1时的平均奖励曲线

Fig. 4 The average reward curves when the reward function is R1

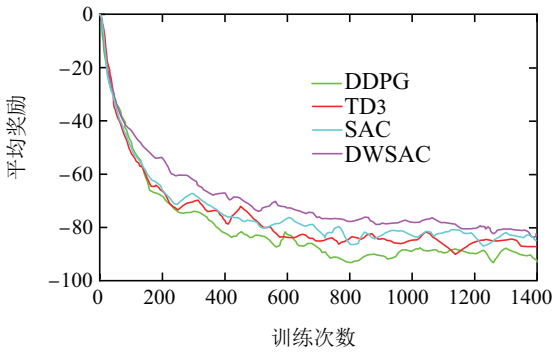



图 5 奖励函数为R2时的平均奖励曲线

Fig. 5 The average reward curves when the reward function is R2

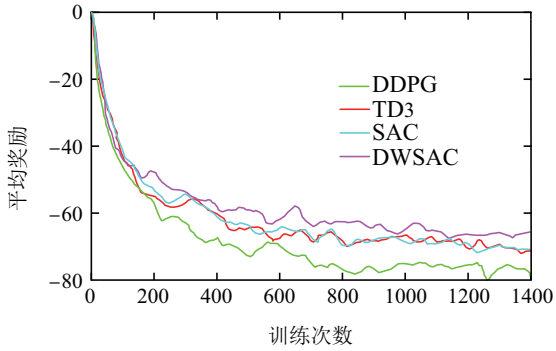



图 6 奖励函数为R3时的平均奖励曲线

Fig. 6 The average reward curves when the reward function is R3

本文采用4种不同的深度强化学习算法用于智能体对交通信号的优化时对平均交叉口通过时间、平均停车次数、平均排队时间、平均队列长度以及队列长度标准差等交通性能指标的影响进行评估, 这里以奖励函数R3为例分析各交通性能指标, 具体数据如表4所示.

从表3中可以看到, 采用DWSAC算法的智能体使车辆的平均交叉口通过时间比采用其他3种算法的平均交叉口通过时间至少提高了10.7%, 平均排队时间的改善率在21.1%到45.7%之间, 平均停车次数的改善率在9.8%到44.7%之间, 平均队列长度的改善率也

提高了19.6%到42.6%, 其标准差的改善率也至少提高了17.4%.

表 4 比较DDPG, TD3, SAC和DWSAC等算法对各交通性能指标的影响及DWSAC相对于其他算法对各性能指标的改善率

Table 4 Compare the impact of algorithms such as DDPG, TD3, SAC and DWSAC on various traffic performance indicators, as well as the improvement rate of DWSAC compared to other algorithms on various performance indicators

| 性能指标 | DDPG | TD3 | SAC | DWSAC |
|---|---|---|---|---|
| 平均交叉口通过时间/s | 87.54 | 78.97 | 67.10 | **59.95** |
| 改善率 | 31.5% | 24.1% | 10.7% | — |
| 平均停车次数/次 | 2.66 | 2.43 | 2.02 | **1.84** |
| 改善率 | 44.7% | 32.0% | 9.8% | — |
| 平均排队时间/s | 56.77 | 48.68 | 39.07 | **30.82** |
| 改善率 | 45.7% | 36.7% | 21.1% | — |
| 平均队列长度/辆 | 9.80 | 8.73 | 7.00 | **5.63** |
| 改善率 | 42.6% | 35.5% | 19.6% | — |
| 队列长度标准差/辆 | 15.54 | 12.91 | 10.42 | **8.61** |
| 改善率 | 44.6% | 33.3% | 17.4% | — |

3) 不同奖励函数对交通性能指标的影响.

DWSAC算法的性能评估已经在前面部分做了详细分析, 因此这里选取DWSAC作为智能体的学习算法来测试当选择不同的奖励函数时, 对交通性能指标有何影响, 具体数据如表5所示.

表 5 在DWSAC算法下对比不同奖励函数对各交通性能指标的影响

Table 5 Comparing the impact of different reward functions on various traffic performance indicators under the DWSAC algorithm

| 性能指标 | R1 | R2 | R3 |
|---|---|---|---|
| 平均交叉口通过时间/s | 59.95 | 74.34 | **57.19** |
| 平均交叉口通过时间标准差/s | 7.22 | 10.56 | **5.43** |
| 平均停车次数/次 | 1.84 | 1.83 | **1.68** |
| 平均停车次数标准差/次 | 0.24 | 0.17 | **0.17** |
| 平均排队时间/s | 30.82 | 45.64 | **28.17** |
| 平均排队时间标准差/s | 7.04 | 10.69 | **5.17** |
| 平均队列长度/辆 | 5.63 | 8.06 | **5.2** |
| 队列长度标准差/辆 | 8.61 | 11.09 | **7.33** |

从表5中数据可以看出, 选取奖励函数R3后智能体的性能更好, 各交通性能指标以及标准差都要好于选择奖励函数R1和奖励函数R2的智能体.

4) 基于累积延迟近似的奖励函数对交通性能指标的影响.

通过上述仿真分析可知, 当采用奖励函数3时, 各项交通性能指标是最好的. 但是, 正如第2.4节提到的, 对于这个奖励函数, 需要分别获取每辆车的延迟来计算交叉口的累计延迟. 在实际情况中, 以目前的检测技术获得每辆车的延迟是不现实的, 所以本文采用了第2.4节中提出的方法来近似计算交叉口的累积延迟. 这里将测试在奖励函数3的基础上使用该近似方法后对交通性能指标有何影响, 具体数据如表6所示.

表 6 在DWSAC算法下基于累积延迟近似的奖励函数3对各交通性能指标的影响

Table 6 The impact of reward function 3 based on cumulative delay approximation on various traffic performance indicators under DWSAC algorithm

| 性能指标 | R3 | 延迟近似R3 | 改善率 |
|---|---|---|---|
| 平均交叉口通过时间/s | 57.19 | 58.97 | −3.1% |
| 平均停车次数/次 | 1.68 | 1.7 | −1.2% |
| 平均排队时间/s | 28.17 | 29.96 | −6.4% |
| 平均队列长度/辆 | 5.2 | 5.51 | −6.0% |
| 队列长度标准差/辆 | 7.33 | 7.64 | −4.2% |

从表6中可以看出, 使用近似算法会导致各交通性能指标有所下降, 但性能损失不大, 重要的是该方法在实际情况中是可行的.

### 4.3 标准连续控制任务下的仿真与分析

第4.2节中的仿真结果表明, 本文提出的DWSAC算法在解决交通信号控制问题时具有良好的性能, 为了进一步测试该算法在控制任务发生变化时是否还具有良好的性能即算法的通用性, 本文从OpenAI gym标准测试集和Humanoid标准测试集中选取了6种具有挑战性的连续控制任务用于测试所提出的算法及其比较算法. 这些用于测试的任务具体包括: Hopper-v2, Walker2d-v2, HalfCheetah-v2, Ant-v2, Humanoid-v2以及Humanoid (rllab), 各标准测试任务的平均奖励曲线如图7所示.

图7中分别显示了DWSAC, SAC, TD3以及DDPG这4种算法在不同连续控制任务中训练的平均奖励曲线. 在每个任务中, 用于测试的4种算法每经过一次完整的训练(episode)执行一次评估, 用于计算平均奖励. 从图7中6张子图的平均奖励曲线可以看出, 无论从算法学习的收敛速度上还是从最终性能上来说, DWSAC在较简单的任务上的表现与主流的3种算法相当, 而在较困难的任务上, 其他3种算法的表现与DWSAC差距较大, 例如, DDPG在Ant-v2, Humanoid-v2和Humanoid (rllab)上基本没有进展, 尤其在后两个任务上毫无进展. TD3在在Humanoid-v2和Humanoid (rllab)

上同样基本没有进展. 从图中曲线的平稳性方面来说, 所提出的算法比SAC略好且远好于TD3和DDPG, 从这方面可以看出, 所提出的算法鲁棒性能更好. 从图中也可以看出, 本文提出的DWSAC在学习速率方面也比原始的SAC算法高. 此外, DWSAC在该仿真中获得的定量结果与文献[18–19]中的算法相比结果也较好, 这表明DWSAC在这些标准任务上的学习效率和最终性能都超过了目前几种主流的算法.

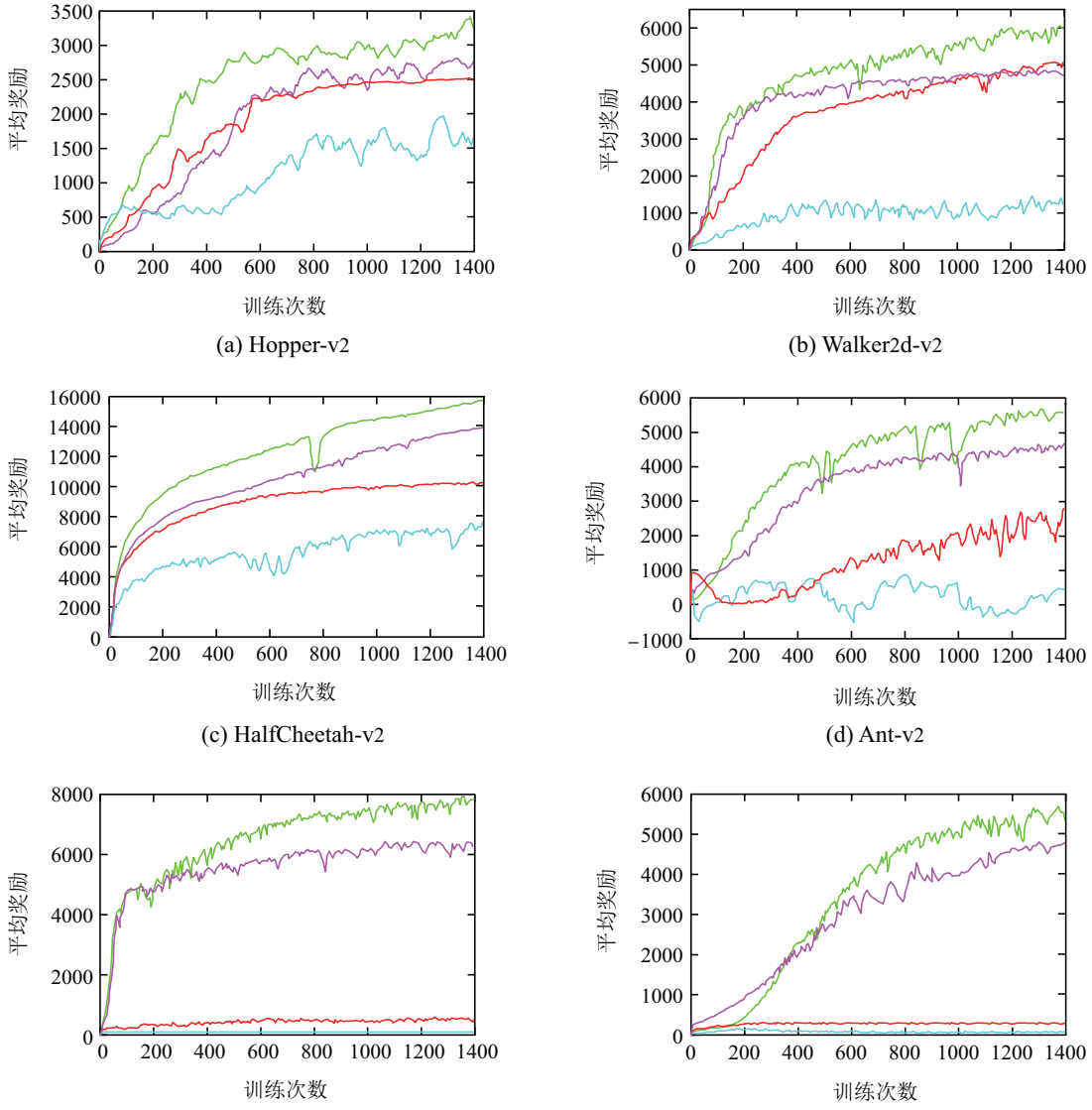



图 7　各标准连续任务上的平均奖励曲线

Fig. 7　Average reward curves on each standard continuous task

## 5　结论

本文研究了基于深度强化学习算法的单交叉口信号控制问题. 首先, 基于可变相序以及考虑有行人穿越交叉口的情况下定义了动作空间. 第二, 分别根据交叉口累积延迟差值的平均变化率、车辆队列长度总和以及交叉口累积延迟差值, 定义了3种奖励函数, 并且在考虑到现实中难以监测所有车辆的延迟之后, 提出了一种延迟近似方法用于近似奖励函数1和奖励函数3的累积延迟. 第三, 在状态空间的定义方面不仅考虑了车辆队列长度还考虑了当前绿灯相位状态和当前绿灯相位经过的时间. 第四, 在深度学习算法方面, 提出了一种基于动态权重的Soft Actor-Critic算法, 该算法面向最大熵强化学习且使用的是随机策略, 在处理具有多个最优动作的问题时相比确定性策略更有优势, 同时引入的动态权重机制在智能体采取的动作有助于系统性能的提高时, 则增强更新范围, 否则减小更新范围, 显著地提高了传统Soft Actor-Critic算法的收敛效率和收敛性能.

在仿真结果方面: 首先, 在交通仿真平台下, 分别测试了 DWSAC, DDPG, TD3, SAC 4种深度强化学习

算法、3种奖励函数、累积延迟近似方法以及速度阈值对平均交叉口通过时间、平均停车次数、平均排队时间、平均队列长度以及队列长度标准差等交通性能指标的影响. 从仿真曲线和具体数据可以看出, 本文提出的模型和DWSAC算法在解决单交叉口信号控制问题上是有效的. 其次, 用4种算法在OpenAI gym标准测试集和Humanoid任务标准测试集中的6个难度不同的任务上做了仿真, 从6个任务的平均奖励曲线可以看出, 所提出的DWSAC算法无论在学习效率、鲁棒性, 还是最终性能方面都优于用于比较的DDPG, TD3和SAC 3种主流的深度强化学习算法.

## 参考文献:


[1] NOAEEN M, NAIK A. Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Systems with Applications*, 2022, 199: 116830.

[2] LI C, HAO W, LU Y. Design of network green bands considering trams. *Journal of Transportation Engineering, Part A: Systems*, 2022, 148(12): 04022108.

[3] FERIANI A, WU D. Multiobjective load balancing for multiband downlink cellular networks: A meta-reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 2022, 40(9): 2614 – 2629.

[4] SAVITHRAMMA R, SUMATHI R, SUDHIRA H S. A comparative analysis of machine learning algorithms in design process of adaptive traffic signal control system. *Journal of Physics: Conference Series*, 2022, 2161(1): 012054.

[5] ZHU Y, YE Y, LIU Y. Cross-area travel time uncertainty estimation from trajectory data: A federated learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 2022, 23(12): 24966 – 24978.

[6] QIAO Z, KE L. Adaptive collaborative optimization of traffic network signal timing based on immune-fireworks algorithm and hierarchical strategy. *Applied Intelligence*, 2021, 51(2): 6951 – 6967.

[7] LIU Y, ZHANG K. Research on platoon dispersion delay of traffic flow considering coordinated control. *Journal of Advanced Transportation*, 2021, 35 (1): 1 – 7.

[8] GENDERS W, RAZAVI S. Evaluating reinforcement learning state representations for adaptive traffic signal control. *Procedia Computer Science*, 2018, 130(8): 26 – 33.

[9] GENDERS W, RAZAVI S. Asynchronous $n$-step Q-learning adaptive traffic signal control. *Journal of Intelligent Transportation Systems*, 2019, 23(4): 319 – 331.

[10] LI L, LV Y, WANG F Y. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 2016, 3(3): 247 – 254.

[11] CHOE C J, BAEK S, WOON B, et al. Deep q learning with lstm for traffic light control. *The 24th Asia-Pacific Conference on Communications*. Ningbo, China: IEEE, 2018: 331 – 336.

[12] COSTA A R, CELIA G R. AC2CD: An actor-critic architecture for community detection in dynamic social networks. *Knowledge-Based Systems*, 2023, 261: 110202.

[13] TRIVEDI P, NANDYALA H. Multi-agent natural actor-critic reinforcement learning algorithms. *Dynamic Games and Applications*, 2023: 13(1): 25 – 55.

[14] TANG X, HUANG B, LIU T. Highway decision-making and motion planning for autonomous driving via soft actor-critic. *IEEE Transactions on Vehicular Technology*, 2022, 71(5): 4706 – 4717.

[15] PENG Y, XIAO L, HEIDERGOOT B, et al. A new likelihood ratio method for training artificial neural networks. *INFORMS Journal on Computing*, 2022, 34(1): 638 – 655.

[16] ZHENG K, JIA X, CHI K, et al. DDPG-based joint time and energy management in ambient backscatter-assisted hybrid underlay CRNs. *IEEE Transactions on Communications*, 2022, 71(1): 441 – 456.

[17] FUJIMOTO S, HOOF H, MEGER D. Addressing function approximation error in actor-critic methods. *International Conference on Machine Learning*. New York: PMLR, 2018: 1587 – 1596.

[18] DUAN Y, CHEN X, HOUTHOOFT R, et al. Benchmarking deep reinforcement learning for continuous control. *International Conference on Machine Learning*. New York: PMLR, 2016: 1329 – 1338.

[19] HENDERSON P, ISLAM R, BACHMAN P, et al. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*. New Orleans: AAAI, 2018, 32: 1531 – 1543.


## 作者简介:


**乔志敏** 讲师, 博士, 目前研究方向为多智能体强化学习和智能交通, E-mail: qiao.miracle@gmail.com;

**柯良军** 教授, 博士生导师, 目前研究方向为多智能体强化学习和群体智能优化, E-mail: keljxjtu@xjtu.edu.cn.