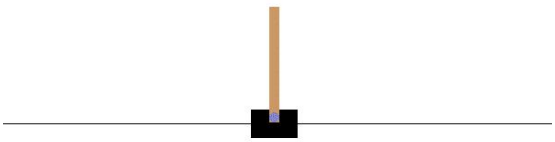
运用深度强化学习方法进行游戏控制

一.问题描述

游戏环境采用了 Gym 的 CartPole、Acrobot、MountainCar、Pendulum、Mountain-CarContinuous、BipedalWalker。前三个环境的动作空间为离散空间，后三个环境的动作空间为连续空间。每个游戏环境的详细介绍如下：

1. CartPole



（1）描述

杆子由一个非驱动的关节连接到小车上，小车沿着无摩擦的轨道移动。钟摆被垂直放置在推车上，目的是通过在推车上向左和向右施力来平衡杆子。

（2）动作空间

动作是一个形状为(1，)的 ndarray，它可以取值{0,1}，表示推车被推的固定力的方向。0 表示向左推小车，1 表示向右推小车。

（3）状态空间

观测值为形状为(4，)的 ndarray，其值对应于以下位置和速度:ndarray[0]为小车的位置，取值范围为-4.8~4.8；ndarray[1]为小车的速度，取值范围为实数空间；ndarray[2]为杆的极角，取值范围为-0.418~0.418 弧度；ndarray[3]为杆的角速度，取值范围为实数空间。

（4）奖励

由于目标是尽可能长时间地保持杆子直立，因此每走一步都会获得奖励， 包括终止步骤，是分配的。奖励阈值为 475。

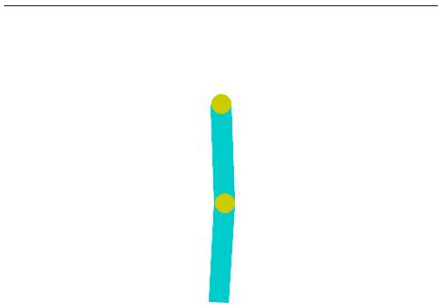
（5）初始状态

所有观测值都分配一个均匀的随机值(-0.05, 0.05)。

（6）结束条件

如果发生以下任一情况，则游戏结束：杆与车之间的极角大于±12°；推车位置大于±2.4（推车中心到达显示屏边缘）；动作序列长度大于 500。

2.Acrobot



（1）描述

系统由两个线性连接形成链条的链节组成，一端链条固定。两个连杆之间的接头被驱动。目标是施加致动接头上的扭矩，用于将线性链条的自由端摆动到上方给定高度。

（2）动作空间

一个 1 维的离散值：0 代表对接头施加一个 0 的扭矩；-1 代表对接头施加一个-1 的扭矩；1 代表对接头施加一个 1 的扭矩。

（3）状态空间

一个 6 维的 ndarray：上部分链条角度的余弦值、正弦值、角速度，下部分链条余弦值、正弦值、角速度。

（3）奖励

目标是使自由端在尽可能少的步长中达到指定的目标高度， 因此，所有未达到目标的步骤都会产生 -1 的奖励。达到目标身高会导致终止，奖励为 0。奖励阈值为-100。

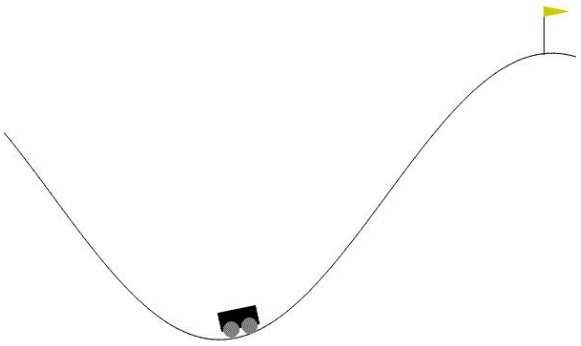
（4）初始状态

两个链条的初始角度设为 0，初始角加速度随机从[-0.1,0.1]取值。

（6）结束条件

当-cos(theta1) - cos(theta2 + theta1) > 1.0 或者一幕的长度超过 500 时，游戏结束。

3.MountainCar



（1）描述

随机放置小车车在正弦谷的底部，唯一的动作是加速度 加速度方向可以是向左也可以是向右。

（2）动作空间

动作是[-1, 1]中一个的连续值，负数代表一个向左的加速度，正数代表一个向右的加速度。

（3）状态空间

状态空间是一个 ndarray(2,)，ndarray[0]表示小车沿 X 轴的位置，ndarray[1]表示小车的速度。各自的取值范围都为实数空间。

（4）奖励

每个时间步都会收到-0.1 * action2 的负奖励，以惩罚采取大规模行动的玩家。如果山地车达到了目标，那么该时间步的负奖励就会加上+100 的正奖励。

（5）初始状态

小车的位置在 [-0.6，-0.4] 中取一个随机值，启动速度为 0。

（6）结束条件

当小车沿 X 轴的坐标大于等于 0.5（小车到达山顶棋子出）或一幕的长度到达 999 时游戏结束。

4.Pendulum

（1）描述

该系统由一端连接到固定点的钟摆组成，另一端是自由的。钟摆以随机位置开始，目标是在自由端施加扭矩以摆动它 进入直立位置，重心正好在固定点上方。

（2）动作空间

动作为施加在自由端的的连续力矩(-2~2)。

（3）状态空间

状态是一个三维的 ndarray：钟摆角度的正弦值(-1~1)，钟摆角度的余弦值（-1~1），钟摆角速度(-8~8)。

（4）奖励

奖励的定义公式为：$r = -(theta^2 + 0.1 * theta\_dt^2 + 0.001 * torque^2)$

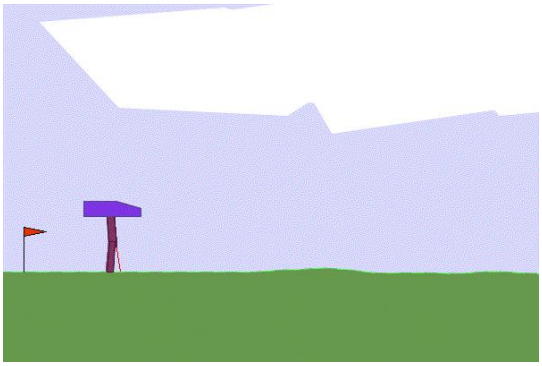
（5）初始状态

钟摆取一个[-pi,pi]随机初始角度，一个[-1,1]的一个随机角速度。

（6）结束条件

一幕的长度达到 200。

5.BipedalWalker



（1）描述

一个简单的 4 关节助行器机器人环境。需要在 2000 个时间步内获得 300 分。

（2）动作空间

动作是多个 [-1，1] 范围内的电机速度值，臀部和膝盖各有 4 个关节。

（3）状态空间

状态由船体角速度、角速度、水平速度、垂直速度，关节位置和关节角速度，腿接触带地面和 10 个激光雷达测距仪测量值。

（4）奖励

前进的奖励，总计 300+积分到远端。 如果机器人跌倒，它会得到-100。施加电机扭矩的成本很小 积分数量。更优化的代理将获得更好的分数。

（5）初始状态

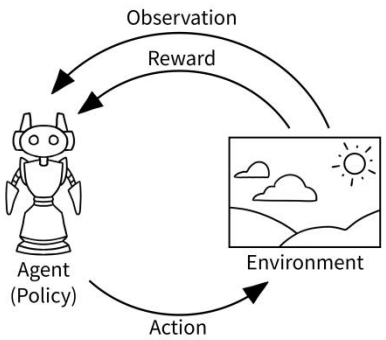
步行者开始站在地形的左端与船体 水平，双腿处于同一位置，膝盖角度轻微。

（6）结束条件

船体与地面接触或步行者超过地形长度的右端。

二.分析

强化学习在解决马尔可夫决策问题上有很好的效果。由于基于表格的强化学习方法存在需要记录的数据量巨大，存储代价高的问题，后来出现了基于函数逼近的强化学习方法。近些年来，深度学习发展迅速并且在很多问题上取得了良好的效果，于是有一些研究工作将基于函数逼近的强化学习方法中的逼近函数用神经网络来拟合，便出现了深度强化学习方法，如 DQN、A2C、PPO、DDPG、SAC 等。



对游戏进行操作属于一个马尔科夫决策过程，适合应用强化学习方法进行求解。

三.实验设计

1. 游戏环境

实验游戏平台采用 Gym（https://www.gymlibrary.dev/content/basic_usage/）。该仿真平台的游戏环境有 6 个系列，分别为 Atari、MuJoCo、Toy Text、Classic Control、Box2D、Third Part Environments。选取了 Box2D 系列下的 Bipedal Walker 和 Classic Control 下的 Acrobot、Cart Pole、Pendulum、Mountain Car Continuous 这五个游戏环境。

2. 算法选用

CartPole、Acrobot、MountainCar 这三个游戏的控制动作为离散值，适合应用 DQN、A2C、PPO 算法进行优化。Pendulum、MountainCarContinuous、BipedalWalker 这三个游戏的控制动作为连续值，适合应用 A2C、PPO、DDPG、SAC 算法进行优化。DQN 算法的估值函数采用三层 MLP 网络拟合。A2C、PPO、DDPG、SAC 这四种算法的 Actor 和 Critic 部分都采用三层 MLP 网络。

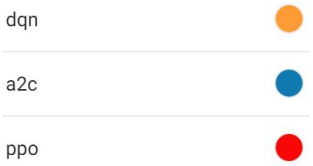
3. 迭代步数

Acrobot、Cart Pole 这个游戏环境的优化算法迭代步数设置为 20k 步。Pendulum、Mountain Car Continuous、Bipedal Walker 这三个游戏环境的优化算法迭代步数设置为 100k。

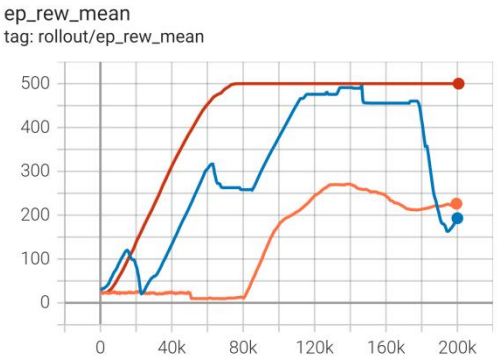
四.实验结果及分析

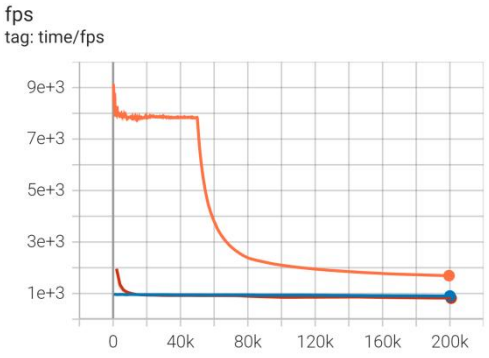
1. CartPole

（1）各个算法曲线颜色：



（2）各个算法的迭代结果及分析：





左图为每幕平均奖励随着更新步数变化的曲线图，可看出 DQN 算法在前 80k 步每幕平均奖励保持在-15 左右，整体无明显的变化趋势，80k~130k 步值呈明显上升趋势，130k~200k 步呈现下降又回升的波动趋势，最后值为 226。A2C 算法前 120k 步的每幕平均奖励呈上升趋势，120k~180k 步值在 460 左右波动，180k 步后呈下降趋势。PPO 算法在前 80k 步，每幕平局奖励一直呈现上升的趋势，到 80k 之后趋于稳定，奖励值稳定在 500。优化过程耗时：DQN 算法耗时 1min10s，A2C 算法耗时 2min，PPO 算法耗时 1min20s。

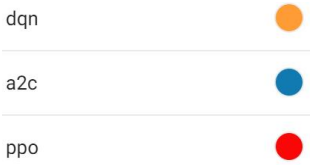
右图为算法每秒迭代次数随着步数的变化曲线图。可看出 DQN 算法每秒的迭代次数在前 1.5k 步呈现下降趋势，从 9100 下降到 7800 左右，1.5k~50k 步保持稳定，50k 步之后呈下降趋势，下降速度逐渐减小，值趋于平稳。A2C 算法每秒的迭代次数保持在 1000 左右。PPO 算法初值每秒的迭代次数为 1959，在前 10k 步逐渐减小，之后保持在 1000 左右。

从实验结果可看出在这个问题上，DQN 算法具有更快的迭代速度，约为 A2C、PPO 算法的两倍，A2C 算法的迭代速度略快于 PPO 算法。优化过程耗时：A2C>PPO>DQN。

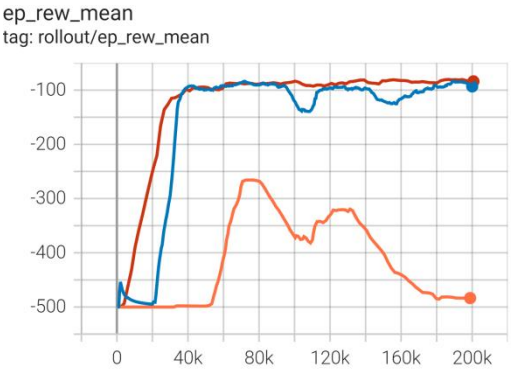
DQN 算法优化效果比较差，只能得到一半最优奖励值的可行解。A2C 算法可以得到与最优解差不多的可行解，但算法不稳定，优化解值稳定一段时间步后，还会出现陡降的问题。PPO 算法更新稳定，在优化值上升或者保持平稳时，曲线都无明显的抖动现象，且可以获得最优解。

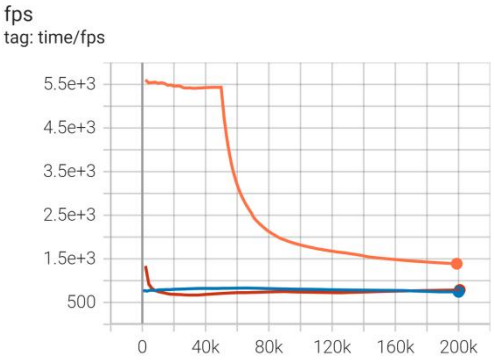
2. Acrobot

（1）各个算法曲线颜色：



（2）各个算法的迭代结果及分析：





左图为每幕平均奖励随着更新步数变化的曲线图，可看出 DQN 算法在前 55k 步每幕平均奖励保持在-500 左右，整体无明显的变化趋势，15k~70k 步值呈明显上升趋势，70k~130k 步呈现下降又回升的波动趋势，130k 步值呈下降趋势。A2C 算法前 40k 步的每幕平均奖励呈现明显的上升趋势，到 40k 步后值在-100 左右波动。PPO 算法在前 40k 步，每幕平局奖励一直呈现上升的趋势，到 40k 之后趋于稳定，奖励值稳定在-85 左右。优化过程耗时：DQN 算法耗时 34s，A2C 算法耗时 50s，PPO 算法耗时 1min。

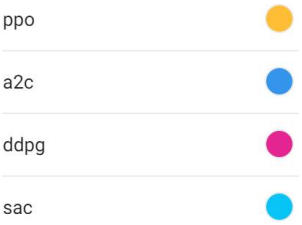
右图为算法每秒迭代次数随着步数的变化曲线图。可看出 DQN 算法每秒的迭代次数在前 45k 步保持在 5500 左右，50k 步后一直呈下降趋势，下降速度逐渐减小。A2C 算法每

秒的迭代次数保持在 750 左右。PPO 算法初值每秒的迭代次数为 1337，在前 15k 步逐渐减小，之后保持在 750 左右。

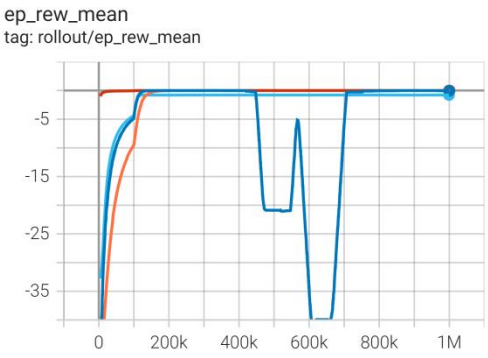
从实验结果可看出在这个问题上，DQN 算法具有更快的迭代速度，约为 A2C、PPO 算法的两倍，PPO 算法的迭代速度略快于 A2C 算法。优化过程耗时：PPO>A2C>DQN。DQN 算法优化效果比较差，其获得的最优可行解只能达到一半左右的优化效果，且该算法不稳定，优化过程完成后会出现优化值波动甚至陡降的现象。A2C 算法可以得到近似最优解，但算法不太稳定，优化过程结束后优化值还会出现一定的波动。PPO 算法更新稳定，在优化值上升或者保持平稳时，曲线都无明显的抖动现象，且可以获得略优于 A2C 的近似最优解。

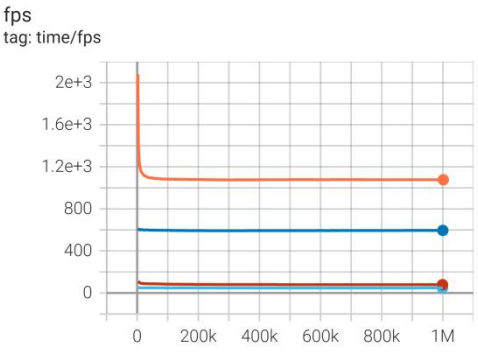
3.MountainCarContinuous

（1）各个算法对应曲线的颜色：



（2）各个算法的迭代结果及分析：





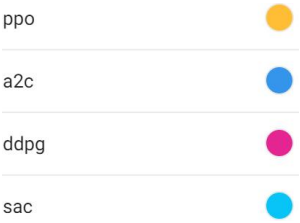
左图为每幕平均奖励随着更新步数变化的曲线图，可看出 A2C 算法在前 120k 步每幕平均奖励整体呈现上升趋势，120k~450k 步值稳定在 0 左右，450k~700k 步呈现较大的波动趋势，700k 步值稳定在 0 左右。DDPG 算法前 20k 步的每幕平均奖励呈现明显的上升趋势，到 20k 步后稳定到 0 左右。PPO 算法在前 140k 步，每幕平局奖励一直呈现上升的趋势，到 140k 之后趋于稳定，奖励值稳定在 0 左右。SAC 算法每幕平均奖励在前 120k 步，每幕平局奖励一直呈现同 A2C 差不多的上升趋势，到 140k 之后趋于稳定，奖励值稳定在 0 左右。优化过程耗时：A2C 算法耗时 3min30s，DDPG 算法耗时 3min，PPO 算法耗时 2min30s，SAC 算法耗时 40min。

右图为算法每秒迭代次数随着步数的变化曲线图。可看出 A2C 算法每秒的迭代次数保持在 600 左右。DDPG 算法每秒的迭代次数保持在 80 左右。PPO 算法初值每秒的迭代次数为 2080，在前 25k 步逐渐减小，之后保持在 1080 左右。SAC 算法每秒的迭代次数保持在 50 左右。

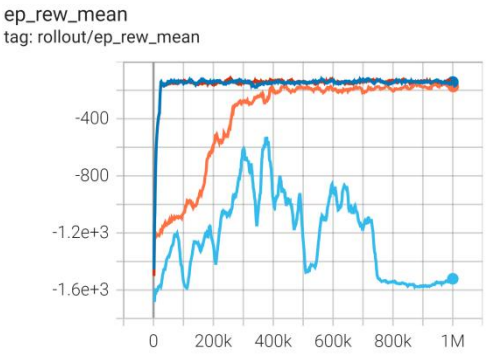
从实验结果可看出在这个问题上，PPO 算法具有更快的迭代速度，约为 A2C 算法的两倍，显著优于 DDPG、SAC 算法；PPO 优化过程耗时最短，SAC 耗时明显长于另外三种算法。四种算法都可以得到近似最优解，而 DDPG 算法的迭代效率高，只需要较少的时间步就可以完成优化。A2C、PPO、SAC 的迭代效率差不多，完成优化过程后，A2C 不够稳定，还会出现优化值陡变的现象。

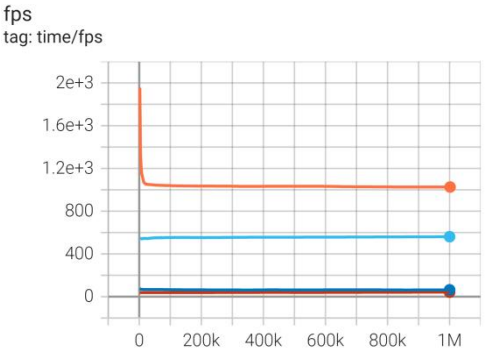
4.Pendulum

（1）各个算法曲线颜色：



（2）各个算法的迭代结果及分析：





左图为每幕平均奖励随着更新步数变化的曲线图，可看出 A2C 算法在前 380k 步每幕平均奖励整体呈现上升趋势，380k~450k 步呈现下降趋势，450k~680k 步又呈现上升趋势，680k~740k 步呈现下降趋势，740k 步值稳定在-1600 左右。DDPG 算法前 30k 步的每幕平均奖励呈现明显的上升趋势，到 30k 步后稳定到-470 左右。PPO 算法在前 400k 步，每幕平局奖励一直呈现上升的趋势，到 400k 之后趋于稳定，奖励值稳定在-450 左右。SAC 算法每幕平均奖励变化趋势同 DDPG 算法。优化过程耗时：A2C 算法耗时 9min，DDPG 算法耗时 6min，PPO 算法耗时 6min，SAC 算法耗时 11min。

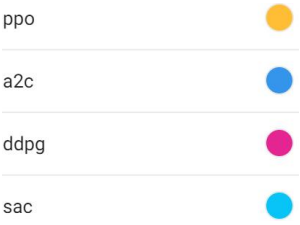
右图为算法每秒迭代次数随着步数的变化曲线图。可看出 A2C 算法每秒的迭代次数保持在 440 左右。DDPG 算法每秒的迭代次数保持在 40 左右。PPO 算法初值每秒的迭代次数为 1955，在前 20k 步逐渐减小，之后保持在 1000 左右。SAC 算法每秒的迭代次数保持在 65 左右。。

从实验结果可看出在这个问题上，PPO 算法具有更快的迭代速度，约为 A2C 算法的两倍，显著优于 DDPG、SAC 算法；优化过程 DDPG、PPO 耗时最短，SAC 次之，A2C 耗时最长。DDPG、PPO、SAC 算法都可以得到近似最优解，而 DDPG、SAC 算法的迭代效

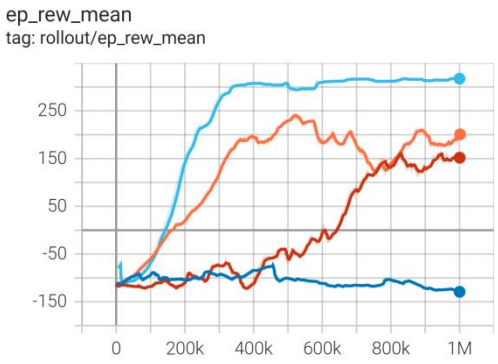
率高，只需要较少的时间步就可以完成优化，PPO 算法需要迭代较多的时间步才能完成优化。A2C 算法迭代不稳定，其最优可行解也只能达到一半左右的优化效果。

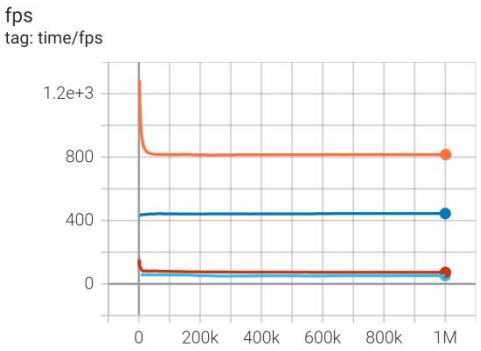
5.BipedalWalker

（1）各个算法曲线颜色：



（2）各个算法的迭代结果及分析：





左图为每幕平均奖励随着更新步数变化的曲线图，可看出 A2C 算法随着迭代，每幕平均奖励一直没有出现明显的增长趋势，稳定在-100 左右。DDPG 算法前 450k 步的每幕平均奖励稳定在-100 左右，而 450k 步之每幕奖励开始增加，到 800k 步稳定到 150 左右。PPO 算法在前 400k 步，每幕平局奖励一直呈现上升的趋势，到 400k 之后趋于稳定，奖励值围绕着 200 波动且波动范围较大。SAC 算法在前 250k 步每幕奖励一直呈上山趋势，到 250k 步之后趋于稳定，奖励值稳定在 300 左右，波动范围较小。优化过程耗时：DDPG 算法耗时 2h50min，PPO 算法耗时 8min，SAC 算法耗时 1h40min。

右图为算法每秒迭代次数随着步数的变化曲线图。可看出 A2C 算法每秒的迭代次数保持在 440 左右。DDPG 算法每秒的迭代次数初始有略微的下降，之后保持在 75 左右。PPO 算法初值每秒的迭代次数为 1400 左右，在前 35k 步逐渐减小，之后保持在 815 左右。SAC 算法每秒的迭代次数保持在 50 左右。

从实验结果可看出在这个问题上，PPO 算法具有更快的迭代速度，约为 A2C 算法的两倍，显著优于 DDPG、SAC 算法；总体来看 PPO 优化过程耗时最短，SAC 虽然迭代速度不如 DDPG 算法，但数据利用效率高，优化过程耗时更短；DDPG、PPO、SAC 可以得到近似最优解。A2C 算法随着迭代时间步增加，优化值无明显变化，无法完成该问题的优化。

五.结论总结

在对离散动作空间的 MDP(马尔可夫决策过程）问题题求解时，DQN 算法拥有最快的迭代速度，但数据利用效率较低，优化过程不稳定，可能会出现网络坍塌，且常常无法获得逼近最优解的可行解。A2C 算法迭代速度速度约为 DQN 算法的一半，数据利用效率较高，优化求解过程较稳定，可以获得逼近最优解的可行解。PPO 算法迭代速度与 A2C 差不多，优化过程稳定，数据利用效率为三种算法中最优，可以获得逼近最优解的可行解。

在对连续动作空间的 MDP 问题求解时，A2C 算法拥有较快的迭代速度，数据利用效率常为四种算法中最差，求解过程波动大对较为复杂的问题进行优化时，算法无法求解。PPO 算法具有最快的迭代速度，优化过程用时最短，优化过程较稳定数据利用效率总体上看要低于 DDPG、SAC，通常可以获得逼近最优解的可行解。DDPG 算法迭代速度与 PPO 算法相比要慢很多，但数据利用效率总体要优于 PPO,有时能达到四种算法中的最优，优化过程稳定，通常可以获得逼近最优解的可行解。SAC 算法迭代速度最慢，数据利用要率总体最优，优化过程很稳定，可以获得逼近最优解的可行解。

总的来看 PPO 算法在对 MDP 问题求解时，有的问题虽然在数据利用效率和优化结果等指标上来看不是最优算法，但 PPO 迭代速度快、优化过程耗时短、优化过程较稳定，没有明显的缺点。故 PPO 算法在很多实际问题上应用很广泛。