

Unknown Title

目录

一.摘要

二.路径规划技术的研究进展

1.研究现状

2.算法分类

 2.1 全局路径规划算法

 2.2 局部路径规划算法

三.本文采用的路径规划算法——强化学习

1. 概念

2. 与其他机器学习方式的区别

3. 强化学习模型

4.马尔可夫决策过程

5. Q-learning算法

四 . 算法设计及代码实现

1.UI设计

2.机器人路径规划算法设计

 2.1 机器人行进方式

 2.2 变量设置：

 2.3 算法步骤

五 . 算法训练结果分析

1.训练参数

 2.训练结果

六 . 实践过程、体会与反思

七. 算法改进和研究展望

1. 基于Q-learning算法改进研究现状

2. 本文算法的后期改进展望

一. 摘要

移动机器人路径规划一直是热门的研究领域，相关的算法丰富多样。本实践前期对机器人路径规划算法做了详细调研，检索并阅读了相关文献，了解了可视图法、人工势场法、启发式算法、神经网络算法等算法在机器人路径规划中的具体应用，本文采用强化学习中的 Q-learning 算法规划机器人的运动路径，做了算法概念学习、算法代码设计、算法参数调优、算法训练测试等具体工作，查阅相关开发资料后，决定应用 QT Creator 5.0.2 作为开发环境，采用栅格建模作为算法应用情景、开发语言为 C++ 语言。

在完成上述算法设计和算法应用情景开发后，对算法进行仿真验证，训练次数分别为 200、1000、5000 次，障碍物比例分别设置为 0.2、0.3、0.5，仿真后导出数据绘制出普通折线图和堆积折线图，呈现结果，也发现了算法收敛速度慢、初始次数搜索效率低等特点。最后对本次实践做出总结和反思，对算法改进的方向做了描述，检索并阅读了大量有关改进强化学习算法规划机器人运动路径的文献，对这些文献一一做了简要概括。并且对本文算法的未来改进计划做出了初步设想。

关键词：移动机器人 路径规划 强化学习 Q-learning

二. 路径规划技术的研究进展

1. 研究现状

路径规划是指在规定区域内规划出一条从起始点到目标点的最优解路径，且要保证与障碍物无碰撞。机器人路径规划存在的难点问题主要有环境建模问题、算法收敛速度慢以及容易陷入局部最优解问题。[1]

2. 算法分类

路径规划可以分为传统算法路径规划和智能仿生算法路径规划两类，其中传统路径规划算法又可以分为全局路径规划算法和局部路径规划算法。

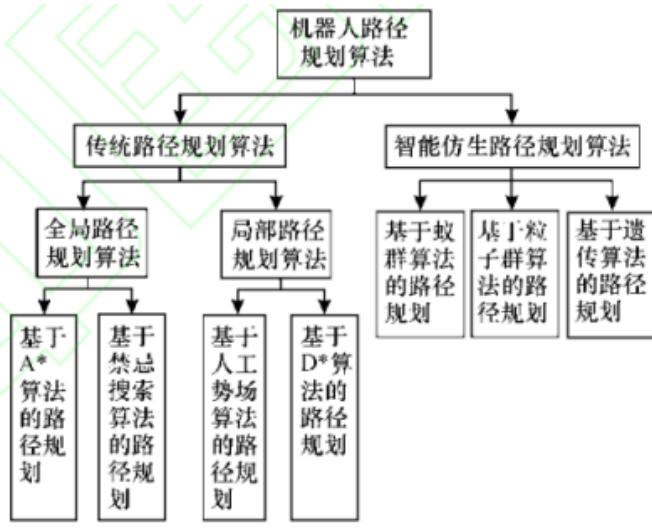


图 2.1 机器人路径规划算法框架图
CSDN @_坐着云起时_

2.1 全局路径规划算法

全局路径规划算法主要包括**可视图法**、**栅格法**和**自由空间法**。

(1) 可视图法

可视图法由 Lozano-Perez 和 Wesley 于 1979 年在论文：《An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles.》中提出。将机器人看作是一个质点，障碍物看作是规则的多边形，分别把初始点、各多边形的各个顶点和目标点之间用直线相连，直线不能穿过障碍物，这样得到的一张图即为可视图。[2] 再经过优化，把一些不必要的连线去掉，由出发点沿着所连的直线就可以到达目标点，该路径就是一条无碰撞的可行路径，这就是可视图法。如图所示。

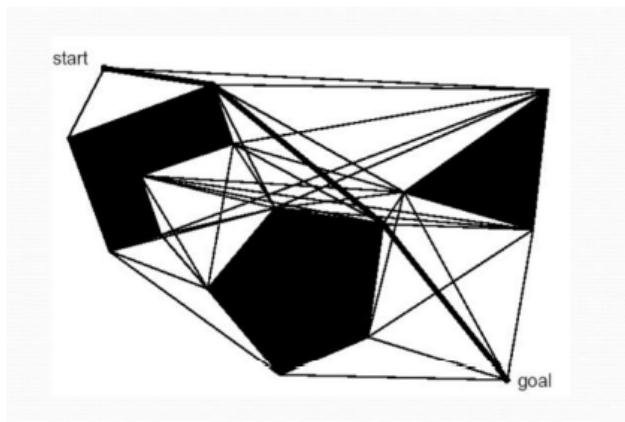


图 2.2 可视图法路径规划
CSDN @_坐着云起时_

(2) 栅格法

采用正方形栅格表示环境，每个栅格有一个表征值 CV，表示在此方格中障碍物对于机器人的危险程度，对于高 CV 值的栅格位置，机器人优先躲避，CV 值按其距机器人的距离被事先划分为若干等级。每个等级对机器人的躲避方向会产生不同影响。

障碍物的位置一旦被确定，就按照一定的衰减方式赋给障碍物本身及其周围栅格一定的值，每个栅格的值代表了该位置有障碍物的可能性。栅格具有简单、实用、操作方便的特点，完全能够满足使用要求。

(3) 自由空间法

主要思想是把障碍物按照一定的比例进行扩大，再把机器人按照合适比例缩小，再用设定好的凸多边形来描述机器人和环境，以此来解决路径规划问题，优点是构建连通图方便，缺点是当障碍物比较多的时候，算法复杂度较高，运行效率低下。

2.2 局部路径规划算法

2.2.1 启发式算法

(1) 遗传算法

Holland 于 1975 年首次提出遗传算法，是模仿生物进化中的基因遗传和优胜劣汰等过程的一种仿生学优化搜索算法。遗传算法通过模拟一个人工种群进化过程，通过**选择(Selection)**、**交叉(Crossover)**以及**变异(Mutation)**等机制，不断迭代搜索出近似最优解，进而达到求解的目的。遗传算法进行搜索时首先要对种群进行初始化，而且需要确定问题求解的编码和解码过程，一般情况下采取实数编码或二进制编码，对于每一次迭代过程，遗传算法需要计算个体的适应度，也就是个体对于问题的匹配程度。适应度的值越大，解的质量就越高。每一次迭代，种群需要经历选择、交叉、变异三个步骤。选择即从种群中选出适应性较好的个体，淘汰适应性较差的个体传入下一代种群。交叉即对个体对中的基因序列进行交换，通过交叉，遗传算法对解的搜索能力得到有效提高。^[3] 变异即对个体基因序列中的基因值进行变动，使得算法具有局部的随机搜索能力，并能够维持群体的多样性，防止算法出现过早收敛的情况。在迭代次数或者种群中最优个体的适应度达到一定阈值后算法终止。种群中适应度最高的个体就是求得的相对最优解。

(2) 粒子群算法

粒子群算法是一种基于群智能的随机优化算法，于1995 年由Kennedy和Eberhart提出。该算法具有高效并行、流程简单、易于编程实现等特点，适于非线性优化问题的求解。其思想来源于对鸟群捕食行为的模拟，优化问题的潜在解通常称为“粒子”，粒子属性可通过位置、速度和适应度函数进行描述。

(3) 蚁群算法

又称蚂蚁算法，是一种用来在图中寻找优化路径的机率型算法。它由Marco Dorigo于1992年在他的博士论文“[Ant system: optimization by a colony of cooperating agents](#)”中提出，其灵感来源于蚂蚁在寻找食物过程中发现路径的行为。蚁群算法是一种模拟进化算法，初步的研究表明该算法具有许多优良的性质。针对 PID控制器参数优化设计问题，将蚁群算法设计的结果与遗传算法设计的结果进行了比较，数值仿真结果表明，蚁群算法具有一种新的模拟进化优化方法的有效性和应用价值。

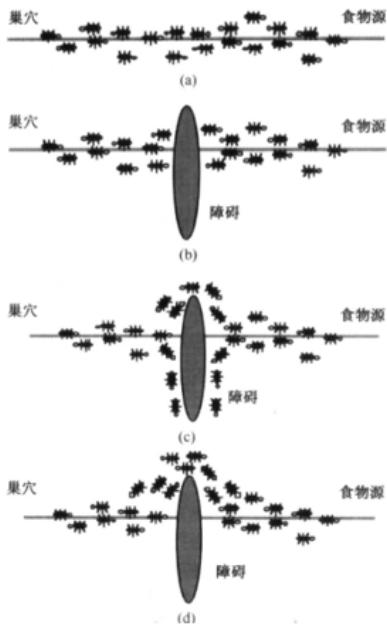


图 2.3 蚁群算法示意图
CSDN @_坐看云起时_

2.2.2 神经网络算法

人工神经网络 (Artificial Neural Networks , ANN) 系统是 20 世纪 40 年代后出现的。它是由众多的神经元可调的连接权值连接而成，具有大规模并行处理、分布式信息存储、良好的自组织自学习能力等特点。BP (Back Propagation) 算法又称为误差 反向传播算法，是人工神经网络中的一种监督式的学习算法。BP 神经网络算法在理论上可以逼近任意函数，基本的结构由非线性变化单元组成，具有很强的非线性映射能力。而且网络的中间层数、各层的处理单元数及网络的学习系数等参数可根据具体情况设定，灵活性很大，在优化、信号处理与模式识别、智能控制、故障诊断等许多领域都有着广泛的应用前景。

2.2.3 人工势场法

人工势场实际上是对机器人运行环境的一种抽象描述。在势场中包含斥力和引力极，不希望机器人进入的区域的障碍物属于斥力极，子目标及建议机器人进入的区域为引力极。引力极和斥力极的周围由势函数产生相应的势场。机器人在势场中具有一定的抽象势能，它的负梯度方向表达了机器人系统所受到抽象力的方向，正是这种抽象力，促使机器人绕过障碍物，朝目标前进。人工势场法规划出的路径一般平滑且安全，结构简单、易于实现，但是有产生局部最优的问题。

三.本文采用的路径规划算法——强化学习

1.概念

强化学习 (Reinforcement Learning ,RL) ，是一种重要的机器学习方法，**机器学习分为监督学习、无监督学习、强化学习**三种。强化学习是系统从环境到行为映射的学习，目的是使奖励信号 (强化信号) 函数值最大。[4] 换句话说，强化学习是一种学习如何从状态映射到行为以使得获取的奖励最大的学习机制。一个动作需要不断在环境中进行实验，环境对动作做出奖励，系统通过环境的奖励不断优化行为，反复实验，延迟奖励。

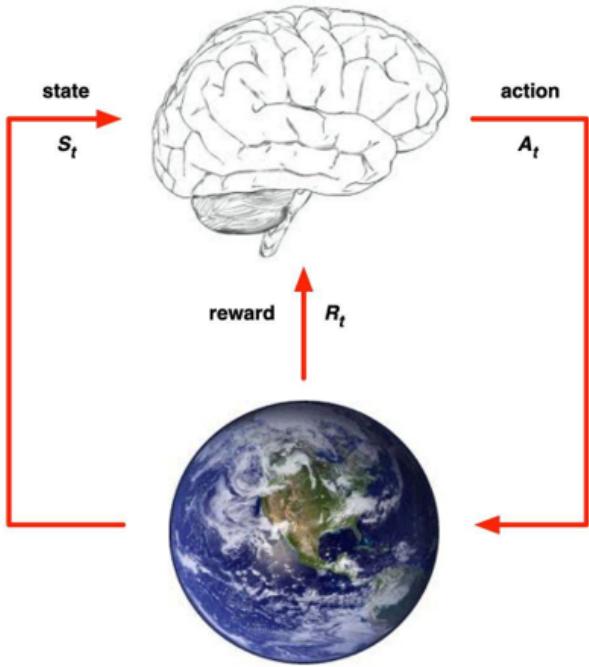
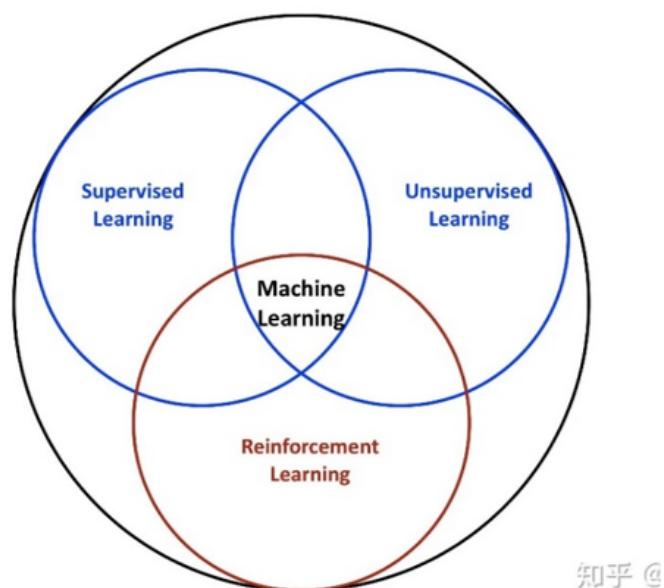


图 3.1 强化学习模型示意图
CSDN @_坐看云起时_

2. 与其他机器学习方式的区别

也就是说与监督学习、无监督学习有什么不同之处。监督学习的训练集中，每一个样本都被打上标签，标签指代的是正确的结果。监督学习让系统按照每个样本所对应的标签推断出应有的反馈机制，进而再没有打过标签的样本上也能计算出正确结果。典型算法有：**决策树、支持向量机（SVM）、k-近邻算法（KNN）等。**无监督学习是从无标签的数据集中发现隐藏的结构，典型的例子就是聚类问题。常见算法有**k-均值(K-means)、DBSCAN密度聚类算法、最大期望算法等。**

强化学习在交互中不存在监督学习中的正确标签，而是在自身的经验中去学习。强化学习的目标也不是寻找隐藏的结构，而是最大化奖励。



知乎 @

图 3.2 机器学习分类及关系
CSDN @_坐看云起时_

3. 强化学习模型

主体(Agent)通过与环境(environment)交互进行学习，**交互包括行动 (action)**，**奖励(reward)**，**状态(state)**。交互过程表述如下：每一步主体都根据策略选择一个行动方式执行，比如说向前走、向后走，然后感知做出该行动后的立即奖励，还有下一步的状态，该状态也就是环境的状态，通过已有的经验再修改自己的策略做出下一个动作，经验就是根据这一步一步的动作学习来的，每一步都积累一些奖励值(立即奖励)，目标就是让累积的奖励值最大。

假设主体所处的环境被描述为状态集S，可以执行的任意的动作集合A。下面描述强化学习模型运作过程：

强化学习系统(在主体上运作)接受环境状态的输入s，根据内部的推理机制，系统给出相应的动作a，环境在系统动作a下由s迁移到新的状态s'，系统再接受环境新状态的输入，同时得到环境对于系统刚才动作的立即奖励r。每次在某状态 s_t 下执行动作 a_t ，主体会收到一个立即奖励 r_t ，环境迁移到新的状态 s_{t+1} 。如此产生了一系列的状态 s_i ，动作 a_i 和立即奖励 r_i 的集合，如图：

强化学习系统的目地是学习一个行为策略 $\Pi : S \rightarrow A$ ，这个策略不是具体的某个动作，更像是一种决策的经验集合，根据这个策略，系统每一步都能做出让环境奖励累积值达到最大的动作。这个环境奖励累积值可以表示为： $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots (0 < \gamma < 1)$ ，其中 γ 为折扣因子，因为主体可能需要若干步才能到达目标，在这个过程中，每一步环境给主体的奖励是要打折扣的。

4. 马尔可夫决策过程

上述对强化学习模型的描述其实就是马尔可夫决策过程，但是对一些值还要详细描述。环境给出的立即奖励 $r_t = r(s_t, a_t)$ ，产生后继状态函数 $s_{t+1} = M(s_t, a_t)$ ，该函数也叫状态转移函数。在马尔可夫决策过程中， $r(s_t, a_t)$ 和 $M(s_t, a_t)$ 只依赖于当前的动作和状态。

主体的任务是学习到策略 $\Pi : S \rightarrow A$ ， $\pi(s_t) = a_t$ 。该策略从初始状态 s_t 开始获得的累积奖励值为：

$$V^\Pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

$$= \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

我们的目的是让上面的 $V^\Pi(s)$ 值最大，则这个策略就是最优策略，表示为

$$\Pi^* = \arg\max V^\Pi(s), (\forall s)$$

CSDN @_坐看云起时_

最优策略 Π^* 的值函数为 $V^{\Pi^*}(s)$ ，记为 $V^*(s)$ 。 $V^*(s)$ 给出了当主体从状态s开始时可获得的最大折算累积回报。

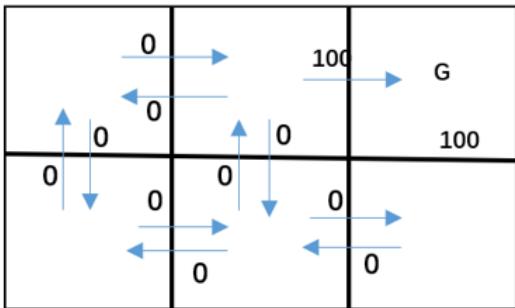


图 3.3 $r(s,a)$ 立即奖励值
CSDN @_坐看云起时_

上图给出一个格状环境，每个箭头代表主体可采取的一个动作，从一个状态转移到另一个。每个箭头关联的数值代表主体执行该动作后的立即奖励 $r(s,a)$ 。状态G可以看成是目标状态，主体只有进入该状态才有大于0的奖励值，进入其他状态的奖励值都为0，一旦主体进入G就只能留在其中。

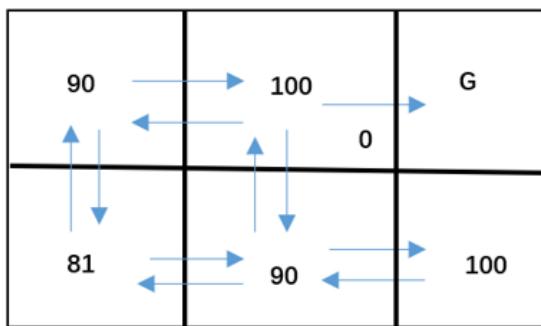


图 3.4 $V^*(s)$ 值
CSDN @_坐看云起时_

如图显示了每个状态的 $V^*(s)$ 值。假设 $\gamma=90$ ，我们上面知道 $V^*(s)$ 给出了当主体从状态 s （该格子）开始时可获得的最大折算累积回报，我们看下方中间的格子，它的 $V^*(s)$ 为90，从该格子走到目标状态的最优决策为先右后上（先上后右也一样），这时可计算出折算累积回报为 $0+\gamma*100+\gamma^2*0+\gamma^3*0+\dots=90$ ，所以此格子用最优决策走到目标状态所能获得的最大累积回报就是90。

5.Q-learning算法

Q-learning是强化学习的算法之一。Q-learning的主要目的就是学习状态动作价值函数的 $Q(s, a)$ ，其中 $Q(s, a)$ 表示的是在给定当前状态 s 和采取动作 a 之后能够获得的收益期望。Q-learning利用状态 s 和动作 a 张成一个Q表来储存Q值，然后根据Q值来选取能够获得最大收益的动作。Q-learning采用是值迭代的方法进行求解，其核心的迭代公式为

$$Q_{k+1}(s, a) = r(s, a) + \gamma \cdot \max_{a' \in A} \{ Q_k(s', a') \}$$

其中 $Q_{k+1}(s, a)$ 是第 $k+1$ 次迭代函数， s 和 a 分别表示的是当前的状态和执行的动作并且它们分别属于状态空间 S 和动作空间 A ， $r(s, a)$ 表示的是在状态 s 执行动作 a 后的即时奖励， s' 和 a' 表示的是下一个的状态和行为， γ 表示的是折扣系数。

算法伪代码如下：

1. 对每个 s, a 初始化表项

2. 观察当前状态 s , 一直重复:

3. 选择一个动作 a 并执行它;

4. 接收到立即回报 r ;

5. 观察新状态 s' ;

6. 对 $Q_{k+1}(s, a)$, 按照 $s \leftarrow s'$ 更新表项 $Q_{k+1}(s, a) \leftarrow r(s, a) + \gamma * \max_{a' \in A} \{Q_k(s', a')\}$

举个例子：

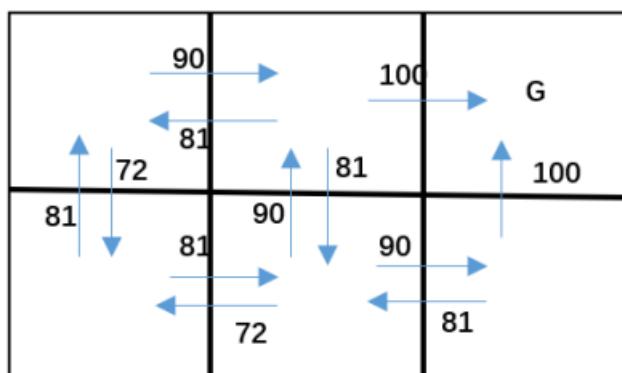


图 3.5 $Q(s, a)$ 值

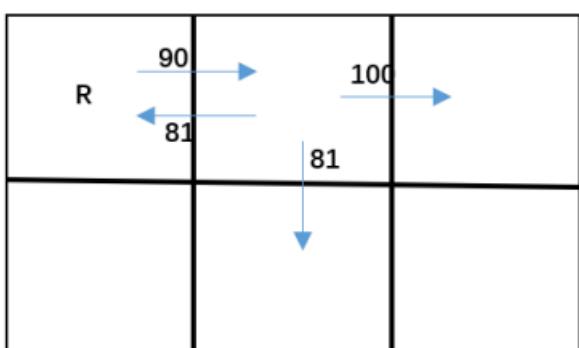


图 3.6 主体的初始状态

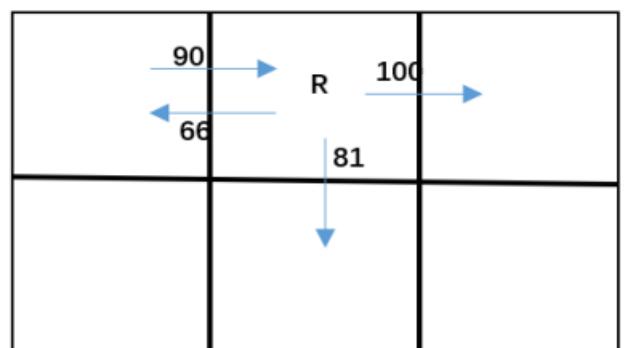


图 3.7 执行单个动作后主体的状态

图3.6是主体的初始状态 s_1 和初始假设中几个相关的Q值，如 $Q(s_1, a_{right})=73$ ，其中 a_{right} 指代主体向右的动作。主体执行动作 a_{right} 后，受到立即奖励为0，并转换到下一状态 s_2 ，然后它基于对新状态 s_2 的Q估计更新其 $Q(s_1, a_{right})$ ，这里 $\gamma=0.9$ 。

$$Q_{k+1}(s,a) \leftarrow r(s,a) + \gamma \max_{a' \in A} \{Q_k(s_2, a')\}$$

$$< -0 + 0.9 \max\{66, 81, 100\}$$

<-90 CSDN @_坐看云起时_

每次主体从旧状态前进到新状态，Q-learning会从新状态到旧状态向后传播其Q值。同时，主体收到的此转换的立即奖励被用于扩大这些传播的Q值，可以证明Q值在训练中永远不会下降。

Q的演化过程如下：初始Q值都为0，算法不会改变任何Q表项，直到它恰好到达目标状态并且收到非零奖励，这导致通向目标状态的转换的Q值被精化；在下一个情节中，如果经过这些与目标状态相邻的状态，那么其非零的Q值会导致与目的相差两步的状态中值的变化；以此类推，最终得到一个Q表。

如果系统是一个确定性的马尔可夫决策过程，则立即奖励值都是有限的，主体选择动作的方式为它无限、频繁地访问所有可能的状态-动作对，那么算法会收敛到一个等于真实Q函数值的Q（近似）。

四 . 算法设计及代码实现

开发环境：Qt Creator 5.0.2 C++语言开发

参考了CSDN博主“榕林子”，“~在下小吴”的文章

程序架构等设计参考了榕林子的文章，链接如下：

文献[5] [人工智能学习之机器人路径规划优化_榕林子的博客-CSDN博客_人工智能路径规划](#)

算法设计参考了~在下小吴的文章，链接如下：

文献[6] [基于Q-learning的无人机三维路径规划（含完整C++代码）_~在下小吴的博客-CSDN博客_无人机三维路径规划](#)

程序架构：

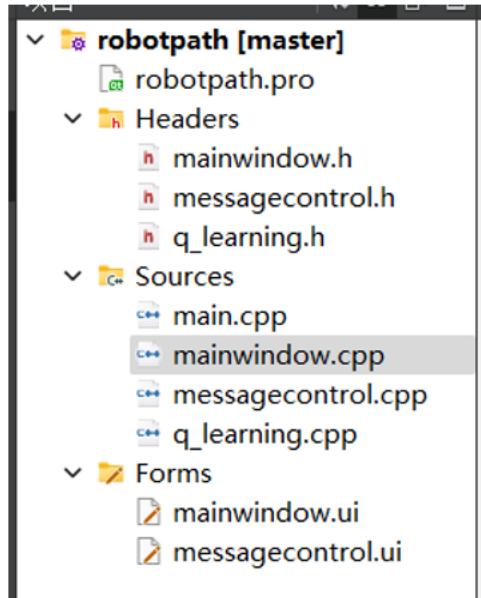


图 4.1 程序架构

Mainwindow中放置了按钮的槽函数，以及强化学习的训练流程，控制的是主页面，Mainwindow.ui也是绘制的主页面，messagecontrol控制的是单元格，它的ui是绘制单元格的效果，messagecontrol.cpp中定义了鼠标事件函数和单元格颜色更改函数，q_learning不需要ui，里面定义了强化学习的action函数，用来控制机器人行动，定义了函数`actpunish()`，用来初始化一些特殊位置的Q值，定义了

`get_expected_max_reward()`，用来返回下一步选择的最大Q值，也就是对应公式中的 $\max_{a^{\wedge} \in A} Q_k(s^{\wedge}, a^{\wedge})$ 。

1.UI设计

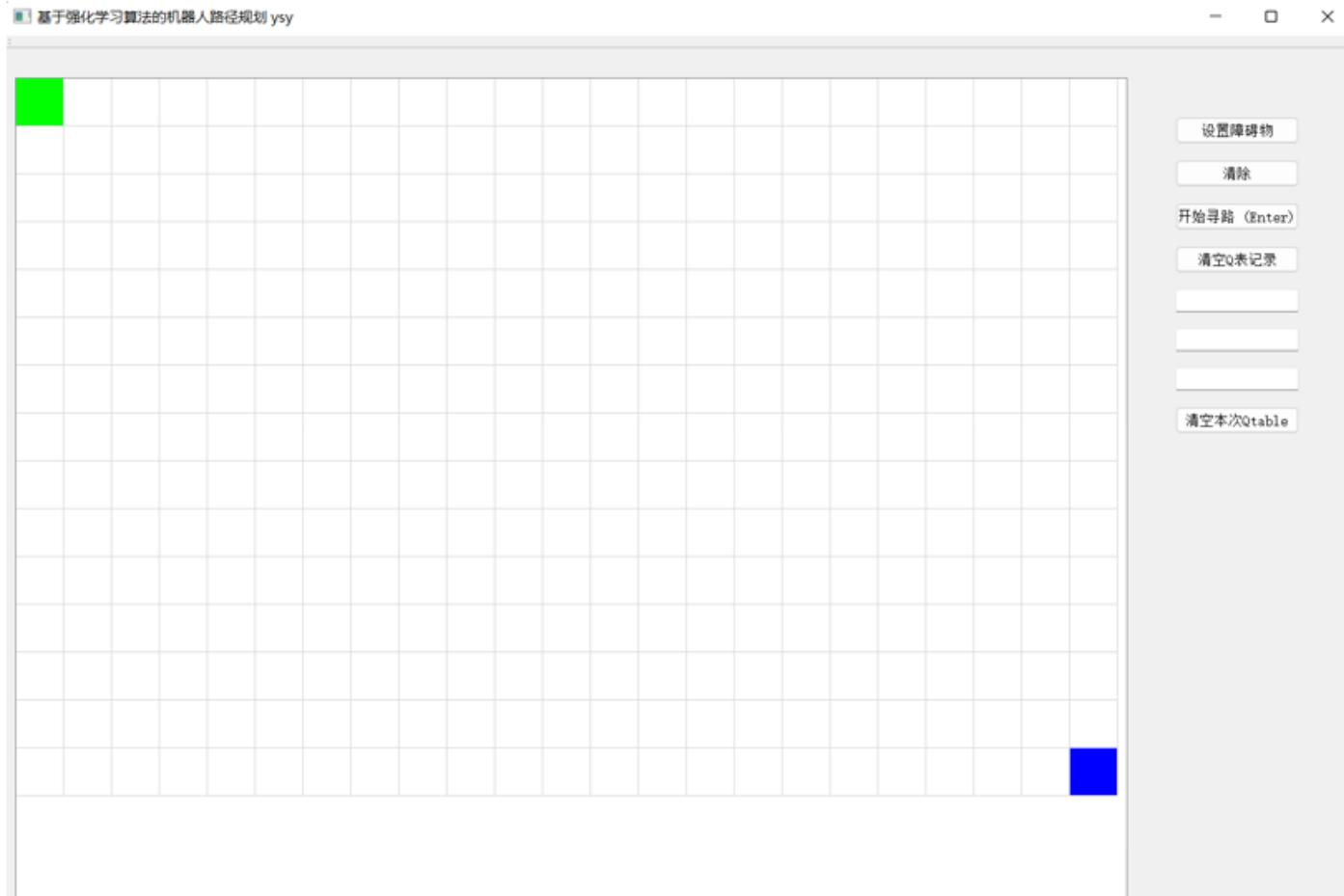


图 4.2 软件主界面

CSDN @_坐看云起_

为机器人设置了行走的空间，以表格的形式展现，这里使用了控件TableWidget,分成23行，15列，但是第一行第一列的坐标都是从0开始，所以最后一个格子（右下角）的坐标是（22,14）。

这里注意，在手动设置UI时，要保证TableWidget足够大，否则在运行程序时，一部分表格会显示不全，我在开发程序的时候，想为起始点（0, 0）设置颜色，弄了很长时间，换了好几种配置颜色的方式，都发现没效果，最后才发现是在.ui那个文件中，TableWidget不够大，遮挡了一部分格子。

画面右侧设置了几何按钮，逐一介绍：

(1) 设置障碍物

鼠标点击单元格，可以选中变色，效果如下：

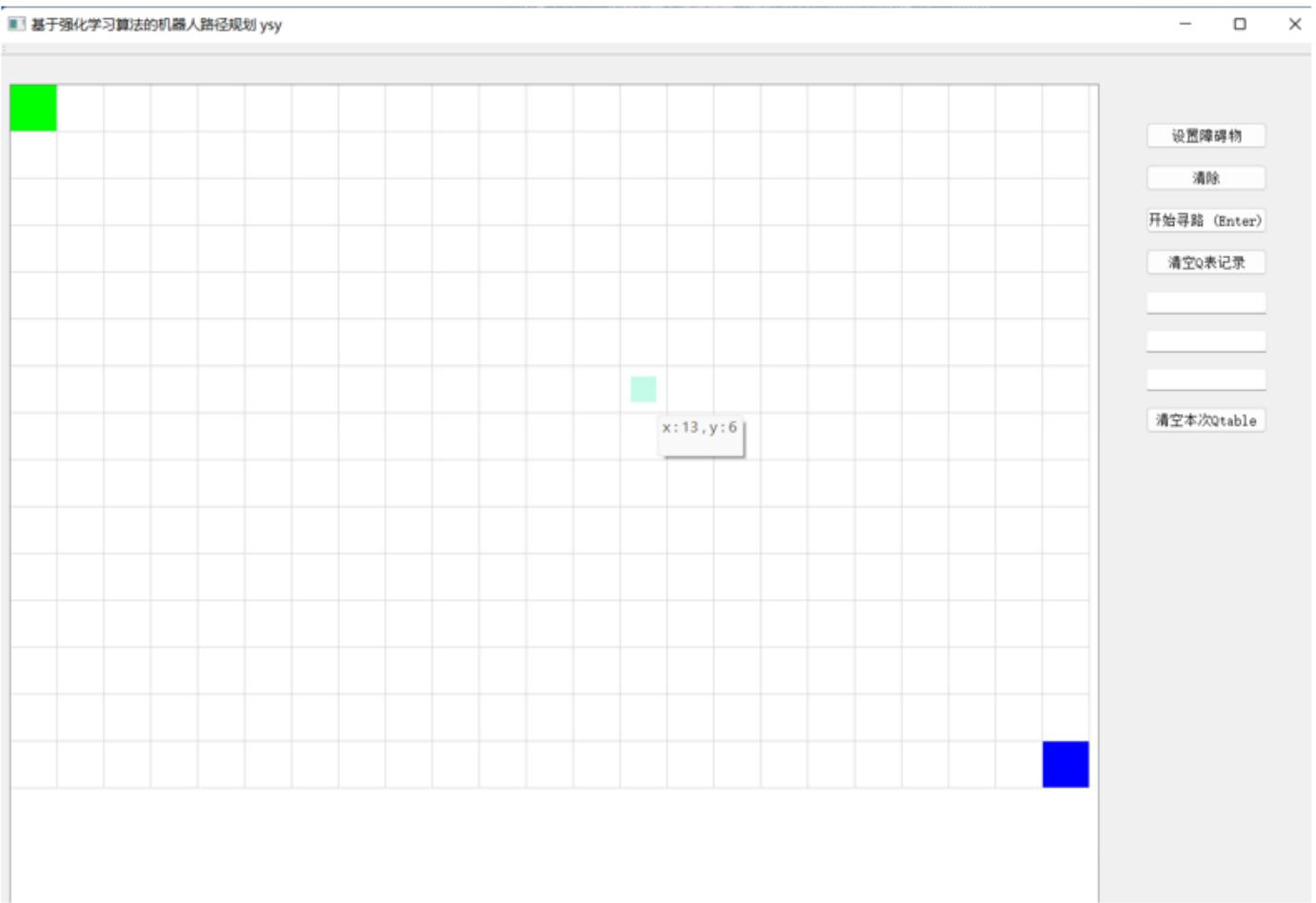


图 4.3 选中的格子效果

CSDN @_坐看云起时_

变色的格子就是鼠标点击选中的，如果后悔不想要选中这个，再点击一次，就可以取消，选中需要设置障碍物的格子之后，点击按钮“设置障碍物”，格子就会变成灰色，效果如下：

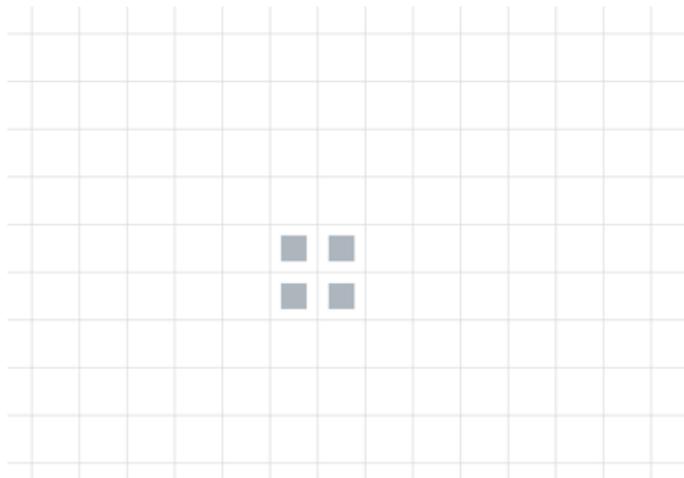


图 4.4 选中为障碍物的效果

CSDN @_坐看云起时_

如果想改变障碍物位置，可以再点击一次格子，会从灰色变成刚才选中的状态，相当于取消了设置成障碍物。

(2) 清除按钮

点击后可以清除所有的障碍物和行走轨迹。

(3) 开始寻路

该按钮点击后，机器人从起始点出发开始寻径，按钮文字会从“开始寻路（Enter）”变为“停止寻路（Enter）”，再次点击按钮，寻径就会暂停，可以随时切换状态。

(4) 清空Q表记录

点击该按钮之后，本地记录Q表值的文本文件就会被清空，可以防止数据冗余，方便后续的算法测试。

此处代码用到Qfile里面的读写

```
1. 1.      QFile file("D:/Qt/project/robotpath/algorithmdata.txt");
2. 2.          //对文件进行写操作
3. 3.          file.open(QIODevice::WriteOnly|QIODevice::Truncate);
4. 4.          file.close();
```

这里面QIODevice::Truncate代表的是打开文件时清空文件所有内容。

(5) 清空Qtable

后面训练要用到这个按钮，否则每次都要重新运行。

(6) 剩余按钮

剩余三个按钮从上至下分别是“打印训练次数”、“打印步数”、“打印成功走到终点的总次数”。用的控件都是LineEdit

“打印训练次数”是当训练次数是10的倍数时才打印出来，“打印步数”也一样。

此处用的代码都是：

```
1. 1.      char buffer1[256];
2. 2.          sprintf(buffer1,"步数为: %d\n",j);
3. 3.          ui->lineEdit_2->setText(buffer1);
```

用sprintf函数把整形变量转化为字符型，存在字符数组中再显示在控件上。

各种点的UI，包括起始点，轨迹，终止点，障碍物，选中点：

先上代码：

```
1. enum CellType
2. 2. {
3. 3.         Null = 0 , Barrier = 1 , Candidate = 2 , Path = 3 ,
Selected=4, Start=5 ,End=6, Win=7
4. 4.     } type;
5. 5.
6. 6.     void MessageControl::setCellType(CellType type)
7. 7. {
8. 8.
9. 9.         this->type = type;
10. 10.     if(type == MessageControl::Null)
11. 11.     {
12. 12.
13. 13.         ui->widget->setStyleSheet (QString ("background-
color:%1;") .arg(defaultColor.name ()));      // 设置背景色
14. 14.
15. 15.     }
16. 16.     if(type == MessageControl::Barrier)
17. 17.     {
18. 18.         ui->widget->setStyleSheet (QString ("background-
color:%1;") .arg(barrierColor.name ()));      // 设置背景色
19. 19.
20. 20.
21. 21.     }
22. 22.     if(type == MessageControl::Candidate)
23. 23.     {
24. 24.         ui->widget->setStyleSheet (QString ("background-
color:%1;") .arg(candidateColor.name ()));      // 设置背景色
```

```
25. 25.  
26. 26.        }  
27. 27.        if(type == MessageControl::Path)  
28. 28.        {  
29. 29.  
30. 30.            ui->widget->setStyleSheet(QString("background-  
color:%1;").arg(pathColor.name()));      // 设置背景色  
31. 31.  
32. 32.        }  
33. 33.        if(type == MessageControl::Selected)  
34. 34.        {  
35. 35.            ui->widget->setStyleSheet(QString("background-  
color:%1;").arg(selectedColor.name()));      // 设置背景色  
36. 36.  
37. 37.        }  
38. 38.        if(type == MessageControl::Start)  
39. 39.        {  
40. 40.            MessageControl::palette.setBrush(QPalette::Active,  
QPalette::Base, QBrush(Qt::green));      // 设置背景色  
41. 41.            this->setPalette(palette);  
42. 42.        }  
43. 43.        if(type == MessageControl::End)  
44. 44.        {  
45. 45.            MessageControl::palette.setBrush(QPalette::Active,  
QPalette::Base, QBrush(Qt::blue));      // 设置背景色  
46. 46.            this->setPalette(palette);  
47. 47.        }  
48. 48.        if(type == MessageControl::Win)
```

```

49. 49.      {
50. 50.          ui->widget->setStyleSheet(QString("background-
color:%1;").arg(winColor.name())); // 设置背景色
51. 51.      }
52. 52.  }

```



思路就是为函数setCellType(CellType type)传进去一个参数，这个参数的数据类型在.h文件中定义过了，是一个元素类型，有起始点(Start)、终止点(End)、障碍物(barrier)、轨迹(Path)等，都可以用数字来表示，传到函数中之后，函数会根据type来设置对应的颜色。那么如何跟单元格选中关联起来呢，如何通过鼠标触发呢？

QT有一个与众不同的“信号和槽”机制，用connect函数可以将两个不同文件中的函数连接在一起，一个函数作为信号发出者，在发送方.h文件中定义为signal，另一个函数作为槽，接收信号，在接收方.h文件中定义为slot。

比如：

发送方： messagecontrol文件

```

31
32     void setCellType(CellType type);
33     int getX();
34     int getY();
35     QPalette palette;
36
37     Q_learning* q_learning;
38
39 signals:
40 |     void selectedCellIndex(int x,int y); //选中栅格
41 |     void selectedCancelIndex(int x,int y); //取消已选中栅格
42 |
43
44
45 protected:
46     void mousePressEvent(QMouseEvent * event); //鼠标事件
47

```

图 4.5 messagecontrol 文件信号函数 CSDN @_坐看云起时_

接收方： mainwindow文件

```

47     int end_y = TABLE_COLUMN;
48
49     void Reinforce();
50
51 public slots:
52     void selectedCellIndex(int x,int y);
53     void selectedCancelIndex(int x,int y);
54
55 private slots:
56     void on_pushButton_barrier_clicked();
57     void on_pushButton_clear_clicked();
58     void on_pushButton_find_clicked();
59     void on_pushButton_clicked();

```

图 4.6 mainwindow 文件槽函数 CSDN @_坐看云起时_

关于QT信号与槽机制具体内容可以查看参考文献[7]。

```

1. 1.      //鼠标控制栅格选择
2. 2.      void MessageControl::mousePressEvent (QMouseEvent *event)
//摁住鼠标事件
3. 3.      {
4. 4.          if(this->type == MessageControl::Selected)           //如
果已经选中了。再次点击及取消选中状态
5. 5.          {
6. 6.              this->setCellType (MessageControl::Null);
7. 7.              emit this->selectedCancelIndex (pos_y,pos_x);    //给
mainwindow类发送一个信号
8. 8.      }
9. 9.      else
10. 10.      {
11. 11.          this->setCellType (MessageControl::Selected);
12. 12.          emit this->selectedCellIndex (pos_y,pos_x);    //给
mainwindow类发送一个信
13. 13.      }
14. 14.      }

```

鼠标选中单元格时，会把该单元格坐标pos_x,pos_y传入信号函数，发送给mainwindow，这里emit就是发送信号的意思，mainwindow中的对应槽函数就会接收到参数，将该单元格坐标加入到已选中坐标数组

中，同时给setCellType传入颜色参数，使其变色。

2.机器人路径规划算法设计

2.1 机器人行进方式

八叉树的方式行进，可以有八个运动方向：

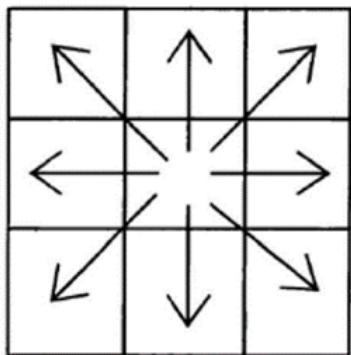


图 4.7 机器人移动策略

用八个常量代表八个方向：

```
1. 1.      const int up=1;
2. 2.      const int down =2;
3. 3.      const int Left =3;
4. 4.      const int Right=4;
5. 5.      const int leftup=5;
6. 6.      const int leftdown=6;
7. 7.      const int rightup=7;
8. 8.      const int rightdown=8;
```

2.2 变量设置：

```
1. 1.      double R=0.8;    //折扣因子
2. 2.      int flag=1;        //用来记录机器人是否位置改变，1为改变
3. 3.      int over;          //判定是否到达终点
4. 4.      double length=0.0; //打印路径长度
```

```

5. 5.      double Qtable [TABLE_COLUMN+2] [TABLE_ROW+2] [8]={0.0}; //存放奖
励函数Q的表，每次训练更新一遍，8(方向)+1(障碍)+4(边沿)=13

6. 6.      double r [TABLE_COLUMN+2] [TABLE_ROW+2]={0.0};

7. 7.      double action(int act , int& x,int& y,int& den);

8. 8.      void initplace(int& x,int& y,int& den);

9. 9.      double get_expected_max_reward(int x,int y);

10. 10.     void actpunish();

```

- R：折扣因子，用来为每一次的延迟奖励打折扣；
- flag: 用来判定机器人目前位置相对于上次的位置是否发生改变，初始设置为1，代表发生改变；
- over: 判断机器人是否到终点的值；
- Qtable: Q值表，每一个单元格有八个对应的Q值，分别代表向一个方向行进的Q值是多少。比如 Qtable[0][0][1]就代表在坐标(0,0)处，向上走的Q值是多少；
- r[]: 立即奖励值，每走到一个单元格，环境都给一个立即奖励；
- Initplace : 初始化，把起始点、终止点、over都初始化；

剩下几个函数前面介绍过了。

！注意：

采用 ϵ - 贪心策略选择动作 a ，在状态 s 下执行当前动作 a ，得到新状态 s' 和奖励 R 。贪心系数Greedy =0.2。Q-learning本质上是贪心算法。但是如果每次都取预期奖励最高的行为去做，那么在训练过程中可能无法探索其他可能的行为，甚至会进入“局部最优”，无法完成游戏。所以，由贪心系数，使得机器人有 Greedy的概率采取最优行为，也有一定概率探索新的路径。

2.3 算法步骤

(1) Action: 机器人从起始点出发，有一定概率随机选择运动方向，否则向Q值最大的方向前进

(2) Reward: 如果遇到边界或障碍，机器人不动，返回一个惩罚值，使该处该动作的Q值减小，下次尽量不做出该动作。

Environment: 如果正常前进，根据做出的动作，由公式

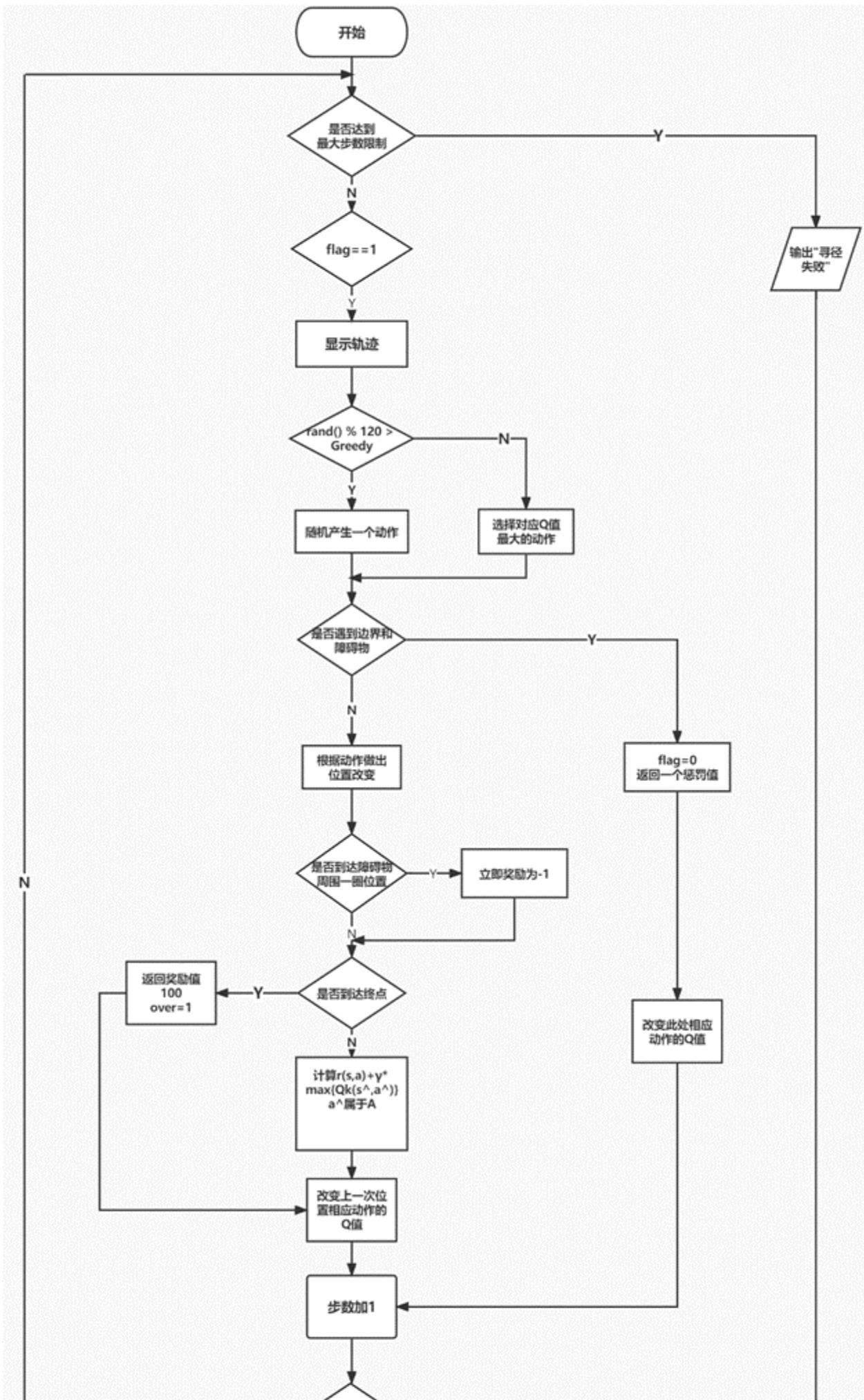
$$Q_{k+1}(s,a) = r(s,a) + \gamma \cdot \max_{a' \in A} \{Q_k(s',a')\}$$

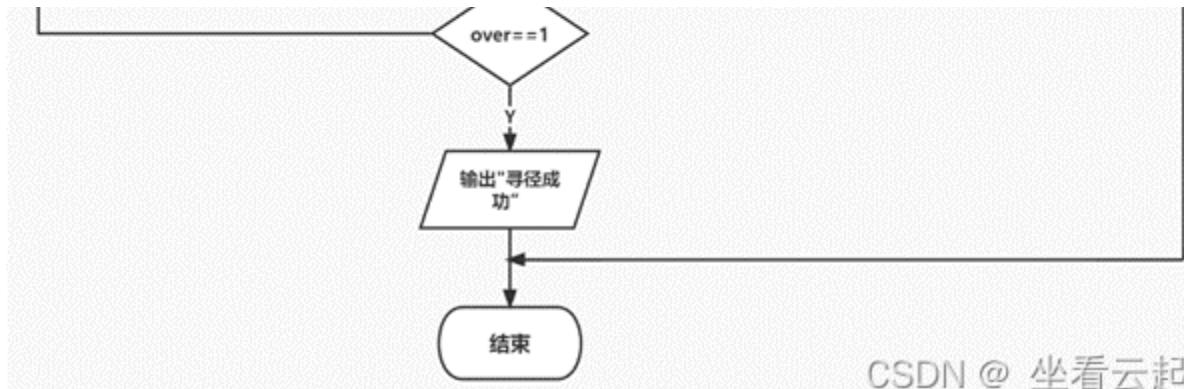
CSDN @坐着云起时

算出上一步所在格子的Q表值。

(4) 超过步数限制 (200步)，则寻径失败。

算法流程图如下 (一次训练) :





CSDN @_坐看云起时_

图 4.8 路径规划算法流程图

算法代码：

```

1. 1.      //强化学习寻径算法
2. 2.      void MainWindow::Reinforce()
3. 3.      {
4. 4.
5. 5.          Q.actpunish();           //初始化边界惩罚值
6. 6.          int wintimes=0;        //定义成功次数变量
7. 7.          int traintimes = QInputDialog::getInt(this,
8. 8.
9. 9.          "QInputDialog_Name",
10. 10.         "请输入要训练的次数",
11. 11.         QLineEdit::Normal,
12. 12.         1,
13. 13.
14. 14.         for(int i=1;i<=traintimes;i++)    //训练循环，traintimes为次
数
15. 15.         {
16. 16.             Q.over=0;                //成功与否判定变量
17. 17.             Q.initplace(start_x,start_y,Q.over);
//初始化，保险

```

```
18. 18.             Q.arrived_xy.clear(); //清空暂存所在格子坐标的数组

19. 19.             Q.arrived_xy.append(QPoint(start_x,start_y)); //把起始点加入暂存所在格子坐标的数组

20. 20.             int op=0; //动作的代号

21. 21.

22. 22.             //每次训练清空前一次的轨迹

23. 23.             for(int i=0;i<=TABLE_ROW;i++)

24. 24.             {

25. 25.                 for(int j=0;j<=TABLE_COLUMN;j++)

26. 26.                 {

27. 27.                     MessageControl* t_MessageControl = (MessageControl*) ui->tableWidget->cellWidget(i,j);

28. 28.                     t_MessageControl->setCellType(MessageControl::Null);

29. 29.                     if(i==0 && j==0)

30. 30.                     t_MessageControl->setCellType(MessageControl::Start);

31. 31.                     if(i==TABLE_ROW && j==TABLE_COLUMN)

32. 32.                     t_MessageControl->setCellType(MessageControl::End);

33. 33.

34. 34.             }

35. 35.         }

36. 36.

37. 37.

38. 38.             //障碍物标记保留

39. 39.             for(auto t:Q.barrier_xy)

40. 40.             {
```

```
41. 41.             MessageControl* t_MessageControl =(MessageControl *)  
    (ui->tableWidget->cellWidget(t.x(),t.y()));  
  
42. 42.             t_MessageControl-  
    >setCellType(MessageControl::Barrier)  
  
43. 43.         }  
  
44. 44.  
  
45. 45.  
  
46. 46.             //当训练次数为10的倍数时，打印在框里  
  
47. 47.             if(i % 10 == 0)  
  
48. 48.             {  
  
49. 49.                 char buffer[256];  
  
50. 50.                 sprintf(buffer,"训练次数为: %d\n",i);  
  
51. 51.                 ui->lineEdit->setText(buffer);  
  
52. 52.             }  
  
53. 53.  
  
54. 54.             //设置一个messagecontrol对象在目前所在格子  
  
55. 55.             MessageControl* t_MessageControl =(MessageControl*) ui-  
    >tableWidget-  
    >cellWidget(Q.arrived_xy.at(0).x(),Q.arrived_xy.at(0).y());  
  
56. 56.  
  
57. 57.             for(int j = 0; j <= 100;j++) //一次训练最大不超过100步  
  
58. 58.             {  
  
59. 59.  
  
60. 60.             //走到10倍数次步数时打印步数  
  
61. 61.             if(j % 10 == 0)  
  
62. 62.             {  
  
63. 63.                 char buffer1[256];  
  
64. 64.                 sprintf(buffer1,"步数为: %d\n",j);
```

```
65. 65.                         ui->lineEdit_2->setText(buffer1);  
66. 66.                     }  
67. 67.  
68. 68.             int regx=Q.arrived_xy.at(0).x(); int  
   regy=Q.arrived_xy.at(0).y(); //提取此时坐标，后面传参要用  
69. 69.             int regtwicex=regx;           int regtwicey=regy;  
//再次暂存此时坐标，后面更新Q表要用  
70. 70.             double nextmax = -100;  
71. 71.  
72. 72.             if(Q.flag==1)  
73. 73.             {  
74. 74.                 MessageControl* t_MessageControl =  
   (MessageControl*) ui->tableWidget-  
>cellWidget(Q.arrived_xy.at(0).x(),Q.arrived_xy.at(0).y());  
75. 75.             //如果位置改变显示轨迹  
76. 76.                 t_MessageControl-  
>setCellType(MessageControl::Path);  
77. 77.  
78. 78.             }  
79. 79.             //有概率随机产生一个动作  
80. 80.             if(rand() % 80 > Greedy)  
81. 81.             {  
82. 82.                 op = rand() % 8 + 1;  
83. 83.             }  
84. 84.  
85. 85.             //如果不随机，寻找Q值最大的动作走  
86. 86.             else  
87. 87.             {  
88. 88.                 for (int m = 1; m <= 8; m++)
```

```

89. 89.          {
90. 90.              nextmax = max(nextmax + 0.0, Q.Qtable[regx]
91. 91.                  [regy] [m]); //遍历各个动作的Qtable值，寻找最大的那个值
92. 92.          }
93. 93.          {
94. 94.          }
95. 95.      if (nextmax == Q.Qtable[regx] [regy] [m])
96. 96.          //寻找Q值最大的那个动作
97. 97.          op = m;
98. 98.      }
99. 99.      int reward = Q.action(op, regx ,regy, Q.over); //做出动作，并存下获得的奖励值
100. 100.

101. 101.      Q.Qtable[regtwicex] [regtwicey] [op] += reward;
102. 102.

103. 103.      if (Q.over == 1)
104. 104.      {
105. 105.          wintimes++; //如果成功，加1
106. 106.

107. 107.      MessageControl* t_MessageControl =
108. 108.          (MessageControl*) ui->tableWidget-
109. 109.          >cellWidget(Q.arrived_xy.at(0).x(),Q.arrived_xy.at(0).y());
110. 110.          t_MessageControl-
111. 111.          >setCellType(MessageControl::Win);

112. 112.          QMessageBox MyBox (QMessageBox::NoIcon, "提
113. 113.          示","正确,寻径成功");

114. 114.          MyBox.exec();

115. 115.          break;

```

```
112. 112. }

113. 113.

114. 114.

115. 115. }

116. 116.         char bufferwintimes[256];

117. 117.             sprintf(bufferwintimes,"成功次数为:
%d\n",wintimes); //展示成功次数

118. 118.             ui->lineEdit_wintimes->setText(bufferwintimes);

119. 119.

120. 120.         if(Q.over != 1)

121. 121.     {

122. 122.             QMessageBox::critical(this,"错误","找不到路
径");

123. 123.         //控件还原

124. 124.             ui->tableView->setEnabled(true);

125. 125.             ui->pushButton_barrier-
>setEnabled(true);

126. 126.             ui->pushButton_clear->setEnabled(true);

127. 127.             ui->lineEdit->setEnabled(true);

128. 128.             ui->pushButton_find->setText("开始寻路
(Enter)");

129. 129.     }

130. 130.

131. 131.

132. 132.         //打印Qtable

133. 133.         //创建文件对象，指定文件位置

134. 134.             QFile
file("D:/Qt/project/robotpath/algorithmdata.txt");

135. 135.         //对文件进行写操作
```

```
136. 136.
    if(!file.open(QIODevice::WriteOnly|QIODevice::Text|QIODevice::Append))

137. 137. {
    138. 138.     QMessageBox::information(this,"警告","请选择正
        确的文件！");
    139. 139. }
    140. 140.     else
    141. 141. {
    142. 142.     QTextStream stream( &file );
    143. 143.     stream<<QString::number(i)<<endl;
    144. 144.     if(Q.over==1)
    145. 145.         stream<<"Win"<<endl<<endl;
    146. 146.     else
    147. 147.         stream<<"fail"<<endl<<endl;
    148. 148.     for(int j=0;j<=TABLE_COLUMN;j++)
    149. {
    150.     for(int i=0;i<=TABLE_ROW;i++)
    151. 149. {
    152. 150.     stream<<QString::number(i)<<" "<<QString::number(j)<<":"<<""
        "; //打印格子坐标
    153. 151.     for(int k=1;k<=8;k++)
    154. 152. {
    155. 153.     stream<<QString::number(Q.Qtable[i][j][k])<<"
        "; //打印Qtable值
    156. 154. }
    157. 155.     stream<<endl;
    158. 156.
    159. 157. }
```

```
160. 158.          }
161. 159.          stream<<endl;
162. 160.      }
163. 161.      }
164. 162.      }
165. 163.
166. 164.
167. 165. void Q_learning::actpunish()
168. 166. {
169. 167.     for(int i=0;i<=TABLE_COLUMN;i++)
170. 168.     {
171. 169.         Qtable[0][i][leftup] = s;
172. 170.         Qtable[0][i][rightup] = s;
173. 171.         Qtable[0][i][up] = s;
174. 172.         Qtable[TABLE_ROW][i][down] = s;
175. 173.         Qtable[TABLE_ROW][i][leftdown] = s;
176. 174.         Qtable[TABLE_ROW][i][rightdown] = s;
177. 175.     }
178. 176.     for(int i=0;i<=TABLE_ROW;i++)
179. 177.     {
180. 178.
181. 179.         Qtable[i][0][Left] = s;
182. 180.         Qtable[i][0][leftup] = s;
183. 181.         Qtable[i][0][leftdown] = s;
184. 182.         Qtable[i][TABLE_COLUMN][Right] = s;
185. 183.         Qtable[i][TABLE_COLUMN][rightup] = s;
186. 184.         Qtable[i][TABLE_COLUMN][rightdown] = s;
```

```

187. 185.      }

188. 186.      Qtable [TABLE_ROW] [TABLE_COLUMN] [Right] = 0;

189. 187.      Qtable [TABLE_ROW] [TABLE_COLUMN] [down] = 0;

190. 188.      Qtable [TABLE_ROW] [TABLE_COLUMN] [rightdown] = 0;

191. 189.

192. 190.      srand (time (0)); //种一个随机种子，后面选择动作要用

193. 191.      }

194. 192.

195. 193.

196. 194.

197. 195.      //判断下一步选择中最大的奖励值

198. 196.      void Q_learning::initplace (int & x, int & y, int & den) //初始化机
器人位置

199. 197.      {

200. 198.      x=0;

201. 199.      y=0;

202. 200.      den=0; //是否到终点，不到都为0;

203. 201.      }

204. 202.      double Q_learning::get_expected_max_reward (int x, int y)
//得到最大期望Q值，用来后面传播到旧状态改变其Q值

205. 203.      {

206. 204.      double Nexstpmxrd = -100;

207. 205.      for (int i=1; i<=8; i++)

208. 206.      {

209. 207.      Nexstpmxrd = max (Nexstpmxrd, Qtable [x] [y] [i]);

210. 208.      }

211. 209.      return Nexstpmxrd;

212. 210.      }

```

```
213. 211.  
214. 212. //行为函数  
215. 213. double Q_learning::action(int act , int& x,int& y,int& den)  
216. 214. {  
217. 215. //边界的惩罚  
218. 216. if((x==0 && act == Left) || (x==0 && act == leftup) || (x==0  
&& act == leftdown))  
219. 217. {  
220. 218.     flag=0;  
221. 219.     return S;           //S为遇到边界和障碍的惩罚值，会在main函数或别  
处全局定义，统一值为-5;  
222. 220. }  
223. 221. if((x==TABLE_ROW && act == Right ) || (x==TABLE_ROW && act  
== rightup) || (x==TABLE_ROW && act == rightdown))  
224. 222. {  
225. 223.     flag=0;  
226. 224.     return S;  
227. 225. }  
228. 226. if((y ==0 && act == up) || (y==0 && act == leftup) || (y==0  
&& act == rightup))  
229. 227. {  
230. 228.     flag=0;  
231. 229.     return S;  
232. 230. }  
233. 231. if((y==TABLE_COLUMN && act == down) || (y==TABLE_COLUMN &&  
act == leftdown) || (y==TABLE_COLUMN && act == rightdown))  
234. 232. {  
235. 233.     flag=0;  
236. 234.     return S;
```

```

237. 235.      }

238. 236.      //障碍物惩罚

239. 237.      if((barrier_xy.contains(QPoint(x+1,y)) && (act==Right || act
    == rightup || act==rightdown)) || (barrier_xy.contains(QPoint(x,y+1))
    && (act==down || act == rightdown || act==leftdown)) ||
    (barrier_xy.contains(QPoint(x,y-1)) && (act== ( up || rightup ||
    leftup) )) || (barrier_xy.contains(QPoint(x-1,y)) && (act==Left || act
    == leftup || act==leftdown))

240. 238.      || (barrier_xy.contains(QPoint(x+1,y+1)) &&
    (act==rightdown)) || (barrier_xy.contains(QPoint(x-1,y+1)) &&
    (act==leftdown)) || (barrier_xy.contains(QPoint(x+1,y-1)) && (act==
    rightup)) || (barrier_xy.contains(QPoint(x-1,y-1)) && (act==leftup
    )))

241. 239.      //格子为障碍物，barrier为vector类型，在其他类中定义，里面存储了障碍物的
    横纵坐标 (points)

242. 240.      {

243. 241.      flag=0;

244. 242.      return S;

245. 243.      }

246. 244.

247. 245.      //其他行为，（正常行走），八叉树

248. 246.      if(act == up)

249. 247.      y--;

250. 248.      if(act == down)

251. 249.      y++;

252. 250.      if(act == Left)

253. 251.      x--;

254. 252.      if(act == Right)

255. 253.      x++;

256. 254.      if(act == leftup)

257. 255.      {

```

```
258. 256.           x--;
259. 257.           y--;
260. 258.       }
261. 259.   if(act == leftdown)
262. 260.   {
263. 261.       x--;
264. 262.       y++;
265. 263.   }
266. 264.   if(act == rightup)
267. 265.   {
268. 266.       x++;
269. 267.       y--;
270. 268.   }
271. 269.   if(act == rightdown)
272. 270.   {
273. 271.       x++;
274. 272.       y++;
275. 273.   }
276. 274.
277. 275.   //走到障碍周围一圈的格子，立即奖励-1，因为贴近障碍，该行为不好
278. 276.   if((barrier_xy.contains(QPoint(x+1,y))) ||
279. 277.           (barrier_xy.contains(QPoint(x,y+1)) ) ||
280. 278.           (barrier_xy.contains(QPoint(x,y-1))) || (barrier_xy.contains(QPoint(x-1,y))) ||
281. 279.           || (barrier_xy.contains(QPoint(x+1,y+1)) ) ||
282. 280.           (barrier_xy.contains(QPoint(x-1,y+1)) ) ||
283. 281.           (barrier_xy.contains(QPoint(x+1,y-1)) && (act== rightup ) ) ||
284. 282.           (barrier_xy.contains(QPoint(x-1,y-1)) ))
285. 283.   //格子为障碍物，barrier为vector类型，在其他类中定义，里面存储了障碍物的
286. 284.   横纵坐标 (points)
```

```

281. 279.    {
282. 280.        r[x][y]=-1;
283. 281.    }
284. 282.
285. 283.        flag=1;
286. 284.
287. 285.        arrived_xy.clear();      //清空旧位置
288. 286.        arrived_xy.append(QPoint(x,y));      //追加新位置
289. 287.        if(x == TABLE_ROW && y == TABLE_COLUMN)      //到达终点
290. 288.    {
291. 289.        den = 1;
292. 290.        return 100;           //终点的奖励;
293. 291.    }
294. 292.        else if(x >= 0 && x < TABLE_ROW && y >= 0 && y <
295. 293.        TABLE_COLUMN)      //正常走，非终点
296. 294.    {
297. 295.        double noractrew = get_expected_max_reward(x,y);
298. 296.        return r[x][y] + R * noractrew;           //计算最大
299. 297.        奖励值
300. 298.    }
301. 299.    }

```

五 . 算法训练结果分析

主要的几个关键量：

(1) 贪心因子greedy：

如果过大，随机选择一个动作的概率更高，机器人不根据Q值做决策，偏向于探索，寻找到终点的速度变慢甚至走不到终点，过小偏向于执行已知的最优策略，易陷入局部最优。这个量的设置问题在强化学习中叫作“探索与利用的平衡问题”

(2) 障碍物比例

障碍物单元格数量占单元格总数量的比重，如果过高，机器人要避障，寻找到终点的速度变慢甚至走不到终点，过小算法收敛过快。

(3) 惩罚值

走到边界的惩罚值与走到障碍物周围一圈的立即回报值。

(4) 折扣因子

设置为**0.8**。

先展示寻径成功与失败的效果：

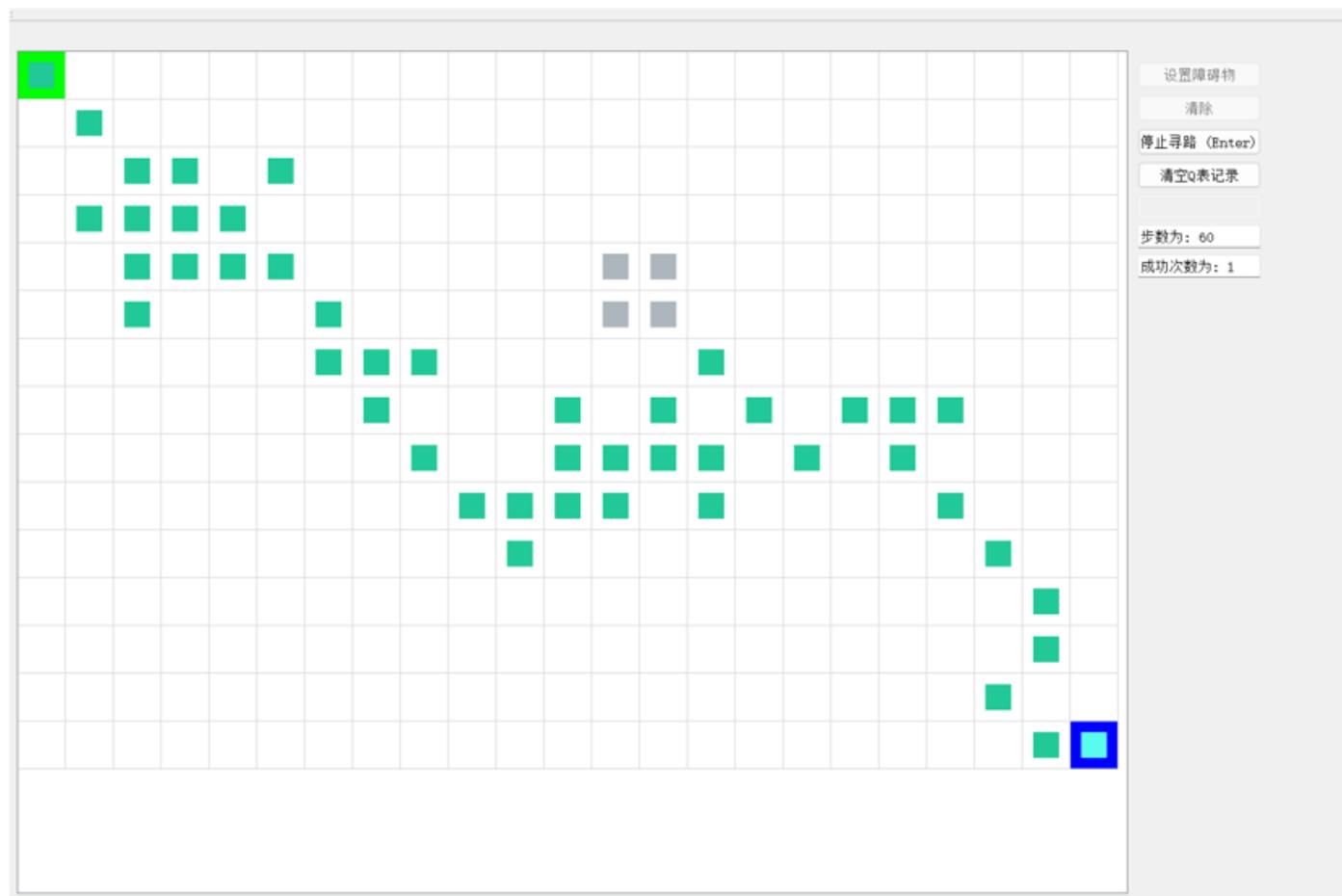


图 4.9 机器人寻径成功示意图

CSDN @_坐看云起时_

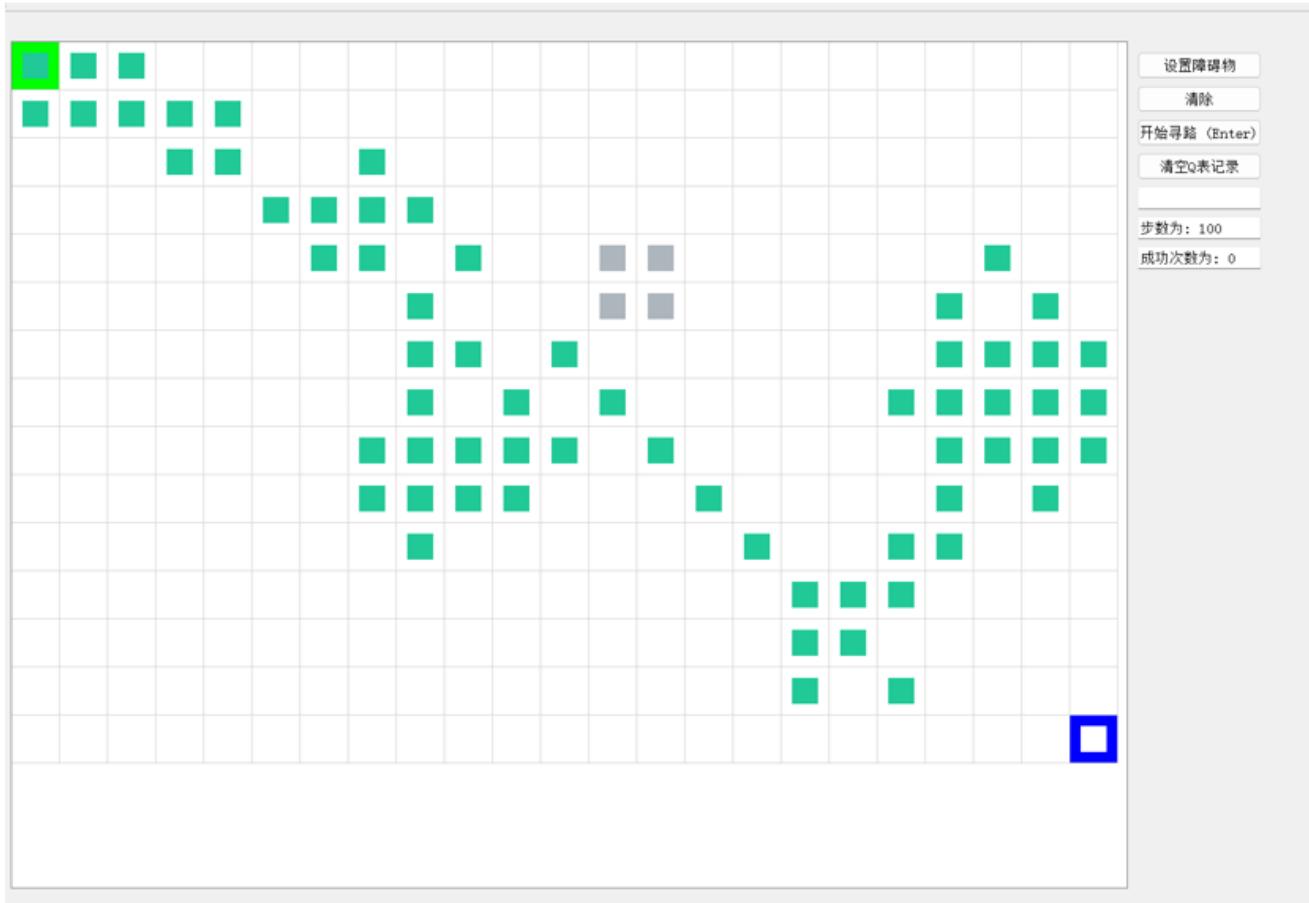


图 4.10 机器人寻径失败示意图

CSDN @_坐看云起时_

1.训练参数

决定训练次数分别设置为**200 , 1000 , 2000**。

每个训练集障碍物比例设置为**0.2 (69块) , 0.3 (104块 (约)) , 0.5 (173块 (约))**

训练结束之后。绘制最短路径长度随成功次数变化的曲线图，并导出最后的Q表。

下面展示障碍物比例分别为0.2、0.3、0.5时的地图：

- **障碍物比例为0.2时：**

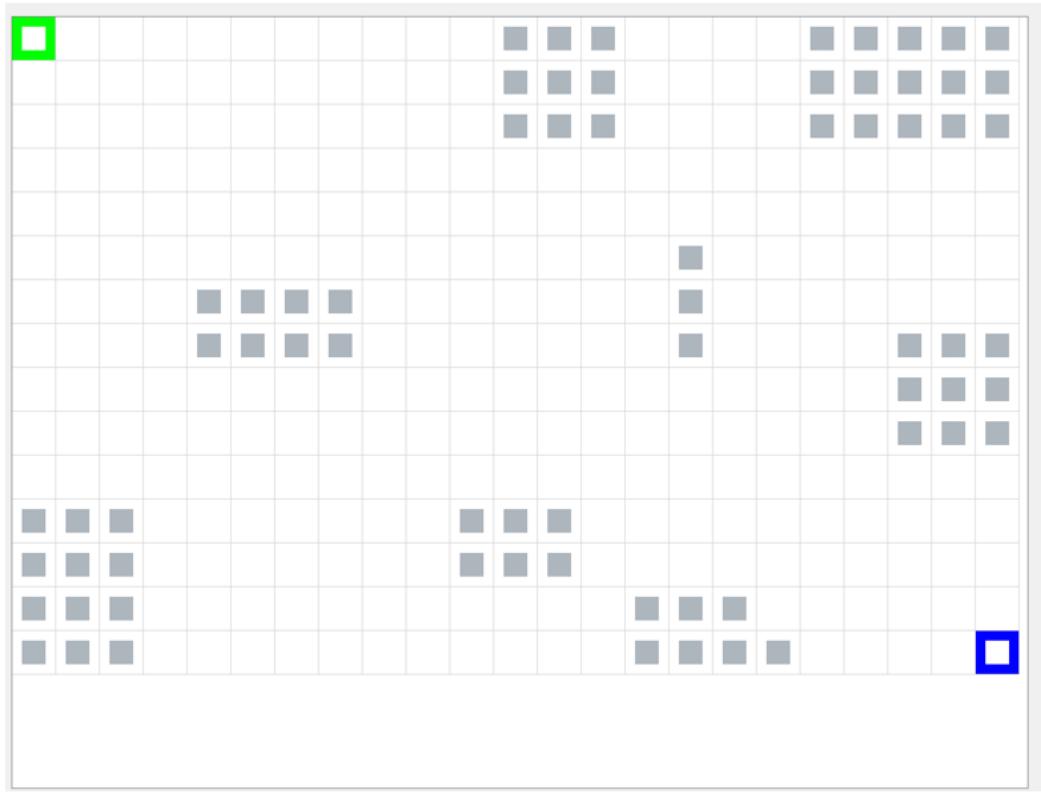


图 4.11 障碍物比例 0.2 时的地图 CSDN @_坐看云起时_

- 障碍物比例为0.3时：

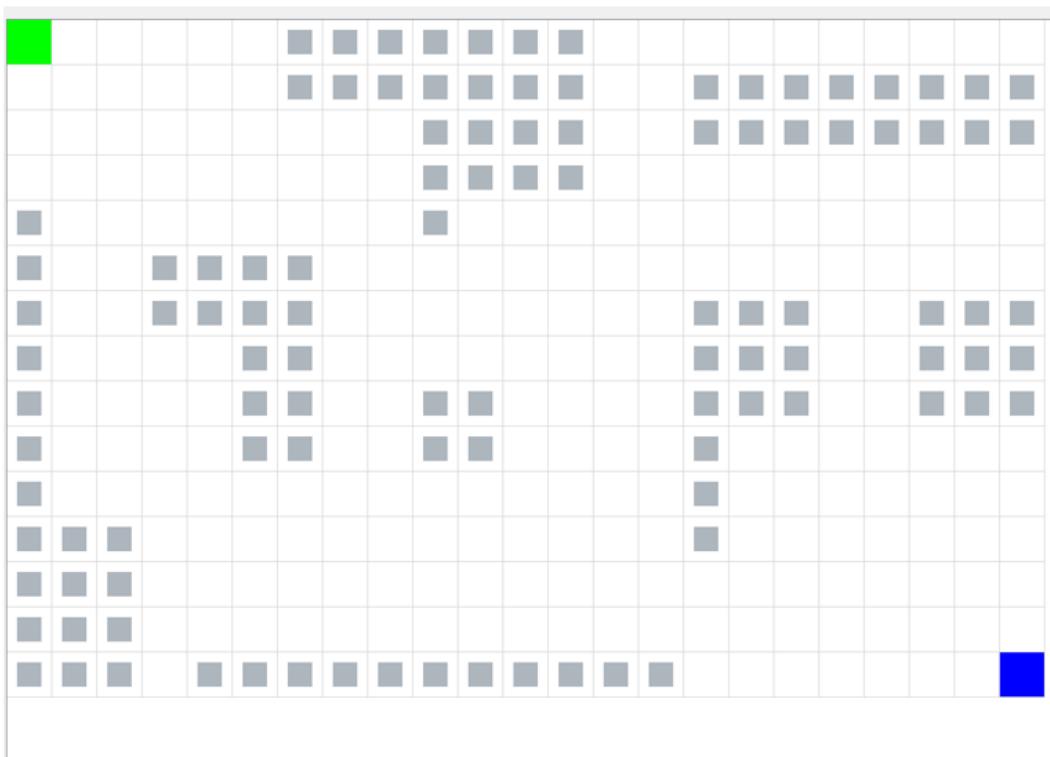


图 4.12 障碍物比例 0.3 时的地图 CSDN @_坐看云起时_

- 障碍物比例为0.5时：

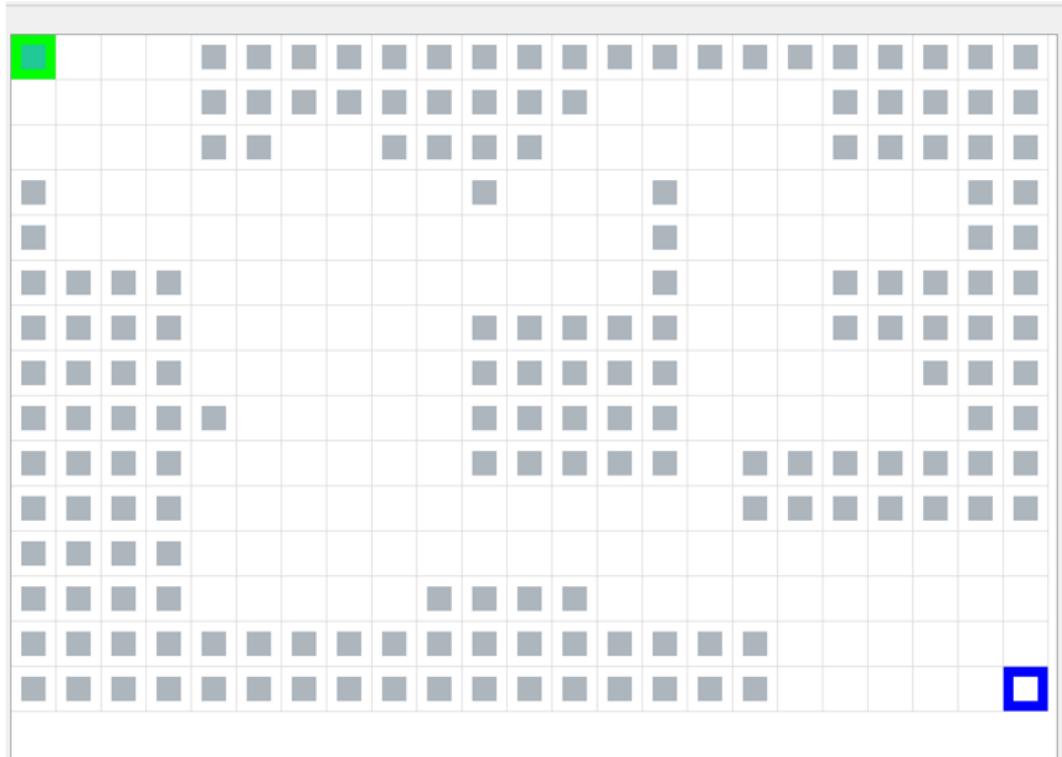


图 4.13 障碍物比例 0.5 时的地图

2.训练结果

(1) 训练次数为200次

- 障碍物比例为20%时

共成功到达目标点15次，算法搜索效率很低，第一次成功规划路径是第2次训练，共耗费88步走到目标点，路径长度为99.0538，但是后面的训练中成功到达目标点时，花费的步数都高于88，且起伏很大，算法并未收敛，而且成功率很低，体现了Q-learning算法的学习效率低的特点。

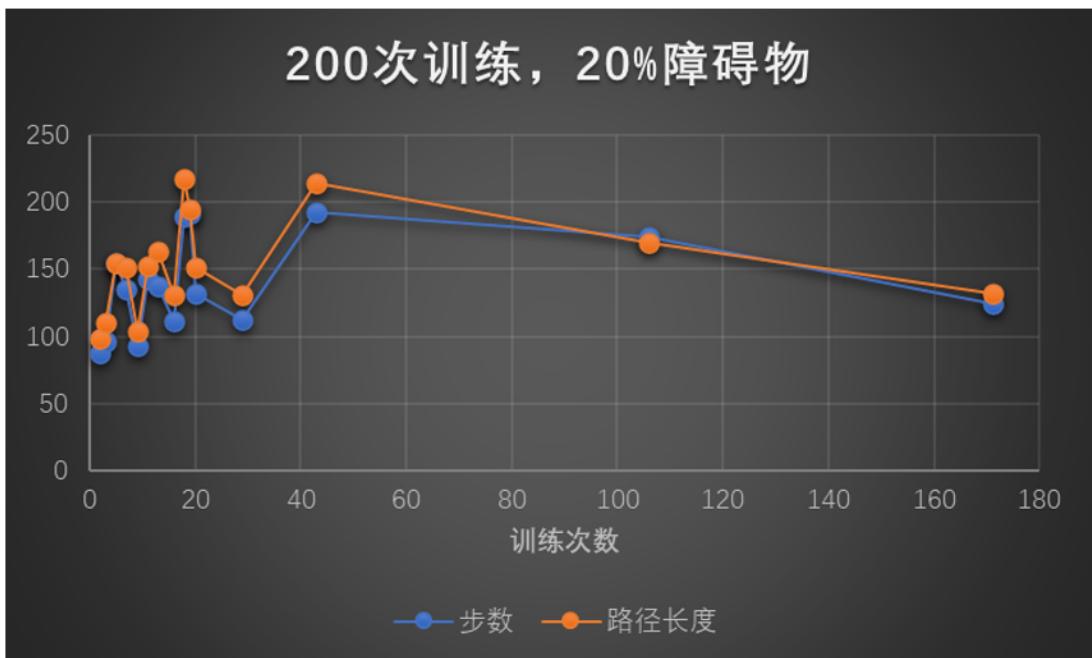


图 4.14 训练 200 次，障碍物比例为 20% 时的步数和路径长度随成功次数变化的图像
CSDN @_坐着云起时_

最后一次寻径成功是第171次，Q表值如下：

每一排的前两个数是横纵坐标，后面8个数分别表示机器人在该坐标处，向上（向前）、下（向后）、左上（左前）、左下（左后）、右上（右前）、右下（右后）行走的Q值。可以观察到训练后期，Q值已经变成位数很大的数据，我发现在训练20次左右的时候，数据就开始突然变大，然后指数倍的增长，不知是什么原因，所产生的数据完全与算法规则不符，突变出大型数值，导致后期规划混乱，算法不收敛，后面的训练都有这种状况。

1. Q表值：

2. 171

3. Win

4. 132.61

5.

6. 0 0: -540 -5.91553e+09 -540 -1.28085e+11 -545 -555 -555
-1.16307e+11

7. 1 0: -195 -6.1787e+10 -8.58994e+09 3.99978e+09 -205 -7.91668e+09
-230 -5.55515e+10

8. 2 0: -545 -1.10888e+11 -1.3556e+11 -1.27496e+10 -525 -8.66903e+10
-535 -1.27769e+11

9. 3 0: -490 -1.02349e+11 -1.15065e+09 -8.76738e+10 -495
-1.32569e+11 -485 -1.37798e+11

10. 4 0: -170 -4.81268e+10 2.99004e+09 -3.59575e+10 -125 -4.24382e+10
 -125 -5.26335e+10

11. 5 0: -110 -4.5614e+10 -2.72063e+10 5.35786e+09 -60 -3.55193e+10
 -105 -3.95195e+10

12. 6 0: -320 -6.43811e+10 -4.29076e+10 -3.53212e+08 -325
 -9.23418e+10 -320 -4.9942e+10

13. 7 0: -205 -2.74853e+10 -3568 -4.23358e+10 -210 -5.29111e+10 -210
 -4.49555e+10

14. 8 0: -95 -2.21518e+10 1.90651e+09 5.46094e+09 -100 -2.11154e+10
 -90 -4.66222e+10

15. 9 0: -355 -1.15964e+11 -1.10417e+11 -7.09791e+10 -355
 -8.54327e+10 -355 -6.67622e+10

16. 10 0: -110 -2.14416e+10 3.57581e+09 -75 -100 -2.62768e+10 -85
 -90

17. 11 0: -5 0 0 0 -5 0 -5 0

18. 12 0: -5 0 0 0 -5 0 -5 0

19. 13 0: -5 0 0 0 -5 0 -5 0

20. 14 0: -5 1.48284e+09 0 0 -5 0 -5 0

21. 15 0: -5 0 -1 2.51721e+08 -10 0 -10 6.71069e+06

22. 16 0: -15 0 2.68428e+06 2.40997e+09 -5 0 -15 -1

23. 17 0: -5 0 0 0 -5 3.94824e+09 -5 -10

24. 18 0: -5 0 0 0 -5 0 -5 0

25. 19 0: -5 0 0 0 -5 0 -5 0

26. 20 0: -5 0 0 0 -5 0 -5 0

27. 21 0: -5 0 0 0 -5 0 -5 0

28. 22 0: -5 0 0 -5 -5 0 -5 -5

29. 0 1: -1.88066e+10 -2.1268e+10 -220 -9.74996e+10 -215 -215
 -6.6572e+10 4.03973e+08

30. 1 1: -1.00372e+11 -6.50058e+09 -7.58131e+09 -6.70237e+10
 -5.83244e+10 -4.43726e+10 -4.46713e+09 6.87142e+09

31. 2 1: -1.84638e+09 4.86814e+09 -7.08167e+10 -5.98703e+10
 -7.45157e+10 -6.13059e+09 2.02657e+09 8.3264e+09

32. 3 1: -4.94324e+09 1.20598e+09 -5.87107e+10 -7.53579e+10
 9.48906e+08 8.83364e+09 -5.82448e+10 -6.8701e+10

33. 4 1: -5.52235e+10 -3.67045e+10 -7.47185e+10 -2.98746e+10
 8.39503e+09 4.07926e+09 -4.58165e+10 -4.22648e+10

34. 5 1: -3.91606e+10 -2.83787e+10 -5.26321e+10 -3.73781e+10
 -2.27742e+10 -3.70459e+10 3.67323e+09 -7.67153e+09

35. 6 1: 4.36288e+09 -8.52213e+08 -3.27019e+10 -2.25191e+10
 -4.14815e+10 -3.08243e+10 2.24586e+09 -3.7972e+09

36. 7 1: 4.91081e+08 -4.11664e+09 -3.00648e+10 -3.37096e+10
 1.56681e+09 1.55488e+07 -2.344e+10 -3.49186e+10

37. 8 1: -3.81581e+10 -2.58559e+10 -3.18528e+10 -5.39406e+10
 1.83811e+09 -2.44116e+09 1.56579e+09 -3.9917e+09

38. 9 1: 4.69681e+09 3.03283e+08 -1.77255e+10 -4.81095e+10
 -4.72446e+10 -3.58218e+10 -2.4228e+10 -3.69386e+09

39. 10 1: -3.09596e+10 -1110 -4.08022e+10 -115 3.16912e+09
 -1.8026e+09 -55 -115

40. 11 1: 0 0 0 0 0 0 0 0

41. 12 1: 0 0 0 0 0 0 0 0

42. 13 1: 0 0 0 0 0 0 0 0

43. 14 1: 0 -3.01819e+08 0 -4.29497e+09 0 -10 5.36855e+06
 -2.93935e+08

44. 15 1: 2.01377e+08 3.41316e+09 1.47653e+09 1.32909e+09 0
 -4.29497e+09 2.14742e+06 -5.75121e+09

45. 16 1: 0 -4.29497e+09 -2.14748e+09 1.38474e+09 5.36855e+06
 -2.14748e+09 1.66136e+09 4.91642e+08

46. 17 1: -1 -5 0 -25 0 -4.16554e+08 -15 -5

47. 18 1: 0 0 0 0 0 0 0 0

48. 19 1: 0 0 0 0 0 0 0 0

49. 20 1: 0 0 0 0 0 0 0 0

50. 21 1: 0 0 0 0 0 0 0 0

```

51. 22 1:  0  0  0  -5  0  0  -5  -5

52. 0 2:  6.36496e+09  -2.31314e+09  -245  -5.3123e+09  -260  -255
   -7.48883e+10  -6.78131e+10

53. 1 2:  -1.24745e+11  -1.21364e+11  -4.66688e+10  -4.30239e+09
   -1.02312e+10  -1.25381e+10  -9.31887e+10  -1.02902e+11

54. 2 2:  -9.51296e+10  -1.20201e+11  -6.02933e+09  -9.2771e+08
   -7.57743e+10  -1.09511e+11  -1.13907e+11  -1.41005e+11

55. 3 2:  -1.52978e+11  -1.08258e+11  -8.72872e+08  -1.14532e+11
   -9.76044e+10  -1.32746e+11  -1.55257e+11  -5.29889e+10

56. 4 2:  -4.10558e+10  -2.97354e+10  3.67449e+09  -4.12405e+10
   -6.04512e+10  -6.0141e+10  -3.91997e+10  -2.25278e+10

57. 5 2:  -2.49379e+10  -4.396e+10  -3.86656e+10  1.29193e+10  -4.29872e+10
   -1.53799e+10  -4.08022e+10  -2.5e+10

58. 6 2:  -5.36871e+10  -2.40415e+09  -4.18071e+10  -2.35989e+10
   -6.15356e+10  -3.4736e+10  -3.33616e+10  -4.56955e+10

59. 7 2:  -8.98349e+10  -1.24688e+11  1.1645e+10  -8.78312e+10  -9.2255e+10
   -1.01789e+11  -8.57611e+10  -1.1112e+11

60. 8 2:  -3.71842e+10  -2.90653e+10  -7.44592e+09  -2.10797e+09
   -3.05268e+10  -4.72446e+10  -3.60639e+10  -3.43597e+10

61. 9 2:  -4.93921e+10  -4.51162e+10  -3.14952e+10  -2.14749e+09
   -4.50989e+10  -2.68088e+10  -3.22123e+10  -4.15921e+10

62. 10 2:  -3.43597e+10  -2.14748e+10  -9.84566e+07  -95  -3.43597e+10
   -3.36084e+10  -105  -105

63. 11 2:  0  0  0  0  0  0  0  0  0

64. 12 2:  0  0  0  0  0  0  0  0  0

65. 13 2:  0  0  0  0  0  0  0  0  0

66. 14 2:  -4  1.90248e+09  -10  7.95864e+09  -20  -10  0  -1.49092e+09

67. 15 2:  -9.36451e+09  -2.14748e+09  -1.62223e+09  -3.18717e+09
   5.28261e+09  -2.23182e+09  0  -3.84657e+09

68. 16 2:  1.7224e+09  -1.03969e+09  5.05899e+09  0  -2.09747e+08
   -2.13131e+09  2.76949e+09  -2.14748e+09

69. 17 2:  -5  3.89958e+08  6.0931e+08  0  0  5.24273e+06  -5  -10

```

70.	18	2:	0	0	0	0	0	0	0	0
71.	19	2:	0	0	0	0	0	0	0	0
72.	20	2:	0	0	0	0	0	0	0	0
73.	21	2:	0	0	0	0	0	0	0	0
74.	22	2:	0	0	0	-5	0	0	-5	-5
75.	0	3:	-1.71966e+10	-5.26933e+10	-255	-5.87267e+10	-260	-250		
			-1.92983e+09	-3.12189e+09						
76.	1	3:	4.55848e+09	-3.28706e+09	-6.36795e+09	-6.87195e+10				
			-2.60479e+10	-4.21974e+10	2.03976e+09	-4.09939e+09				
77.	2	3:	1.14201e+09	-7.47005e+09	-7.82814e+10	-4.72158e+10				
			2.70768e+09	-1.71269e+10	6.03553e+09	-2.12533e+10				
78.	3	3:	3.25452e+09	-4.24381e+10	-7.28441e+10	-3.14588e+10				
			-1.00518e+09	-8.69821e+09	-4.22347e+10	-3.38572e+10				
79.	4	3:	-4.93349e+10	-1.60912e+10	-3.48118e+10	-3.54815e+10				
			3.13446e+08	-1.41097e+10	-1.77848e+10	-3.68379e+10				
80.	5	3:	-3.27198e+10	-1.527e+10	-1.32513e+10	-1.95125e+10				
			-3.77475e+10	-2.73664e+10	1.09849e+10	-3.30701e+10				
81.	6	3:	8.77895e+08	-3.37847e+10	-2.64679e+10	-4.3584e+10				
			-1.82733e+10	-3.9475e+10	-4.29579e+09	-5.13803e+10				
82.	7	3:	-3.56114e+08	-1.94765e+10	-2.18203e+10	-4.71437e+10				
			5.38739e+09	-3.96011e+10	-9.96434e+09	-5.73433e+10				
83.	8	3:	-4.57241e+10	-2.41749e+10	-2.69988e+10	-3.65454e+10				
			-1.23591e+09	-3.49142e+10	-1.84417e+09	3.24535e+08				
84.	9	3:	4.56697e+09	3.91613e+08	-2.57698e+10	-3.22123e+10				
			-2.12587e+10	-2.15689e+10	-4.29497e+09	-2.30489e+10				
85.	10	3:	1.18171e+09	-1.9645e+10	-2.36605e+10	-1.7115e+10	2.76083e+09			
			-56000	-95	-2.78956e+10					
86.	11	3:	-30	-1.50108e+10	-1.07201e+10	3.95163e+09	6.01105e+08			
			2.83836e+09	-55	-1.62417e+10					
87.	12	3:	-85	-1.50324e+10	-1.02519e+10	-1.48733e+10	-55	-2.46344e+10		
			-60	-1.71799e+10						

88. 13 3: -25 -1.50324e+10 2.75357e+09 5.62974e+08 -10 -1.07104e+10
 -2.14748e+09 9.99007e+08

89. 14 3: -3.68566e+09 -5.13718e+08 -4.29497e+09 0 -35 -4.29497e+09
 3.64378e+09 -2.14073e+09

90. 15 3: 3.09311e+09 -2.56404e+09 -7.90863e+08 1.04855e+07 -10
 4.95511e+09 -2.14748e+09 8.32671e+09

91. 16 3: 0 -4.29497e+09 -6.44245e+09 -7.6274e+08 3.34238e+09
 -2.58417e+09 -1 -6.42934e+09

92. 17 3: 4.87448e+08 -2.14748e+09 -2.14748e+09 -2 3.46186e+09 -9
 -20 -1.05014e+09

93. 18 3: 0 0 0 -4 4.19418e+06 0 0 0

94. 19 3: 0 0 -3 0 -10 0 0 0

95. 20 3: -20 0 -9 -10 -5 0 0 0

96. 21 3: 0 0 0 0 -10 0 -25 0

97. 22 3: -5 0 -24 -15 0 0 -5 -10

98. 0 4: -1.40299e+10 -3.42894e+09 -130 4.85291e+09 -95 -115
 -4.1944e+10 -2.472e+10

99. 1 4: -5.32282e+10 -3.72546e+10 -7.00538e+10 -8.12827e+09
 -2.05144e+09 -9.30211e+09 -7.94569e+10 -1.40199e+10

100. 2 4: -5.57748e+10 -1.11064e+10 -1.96384e+09 -3.09155e+10
 -4.23723e+10 -4.16637e+10 -4.67246e+10 -3.27691e+10

101. 3 4: -4.03268e+10 -2.26234e+10 -4.34015e+09 -1.91254e+10
 -3.08729e+10 -5.29545e+09 -3.21019e+10 4.06524e+08

102. 4 4: -3.24975e+10 3.25405e+09 -3.26537e+10 -3.33292e+10
 -5.04557e+10 -3.6537e+10 -5.04513e+10 -4.35873e+10

103. 5 4: -3.188e+10 -5.43024e+09 -1.79122e+10 -2.43432e+10
 -2.99194e+10 -1.88603e+09 -2.39942e+10 1.49662e+10

104. 6 4: -2.0501e+10 4.06111e+09 -4.45431e+10 -3.07393e+10
 -8.75855e+09 2.04694e+09 -3.9516e+10 -2.26481e+10

105. 7 4: -4.07782e+10 -2.35633e+10 -3.29332e+10 -4.66929e+10
 -1.10633e+10 3.89952e+09 -3.69024e+10 2.24423e+10

106. 8 4: -2.92685e+10 3.27585e+10 -3.53616e+10 4.02782e+09
 -2.05767e+10 -2.3333e+10 -2.14748e+10 -2.36223e+10

107. 9 4: -4.10883e+10 -5.30565e+10 -4.30437e+10 -3.96544e+10
 -3.79254e+10 4.43601e+09 -5.6016e+10 -3.43916e+10

108. 10 4: -1.58223e+10 -1.3215e+10 1.51345e+09 -1.5674e+10
 -2.22985e+10 -1.93274e+10 -2.87591e+10 -2.35051e+10

109. 11 4: -7.28426e+09 -1.50324e+10 -2.22712e+10 -2.00556e+10
 -1.71799e+10 -1.09761e+10 4.26131e+09 4.74003e+09

110. 12 4: 1.83358e+10 1.19776e+09 -1.69327e+10 -4.28142e+09
 -2.14316e+10 -1.07374e+10 -6.2834e+09 -6.41688e+09

111. 13 4: -9.81673e+09 -8.58993e+09 -7.55228e+08 4.4763e+09 9.0834e+08
 8.3329e+09 -1 -2.14748e+09

112. 14 4: 1.29728e+09 -5.19378e+08 -8.58993e+09 -1.15677e+09
 -6.44245e+09 -9.73681e+09 -4.40004e+09 -35

113. 15 4: -2.23638e+09 -15 -2.46457e+09 -5.13554e+08 -1.5744e+09 -25
 -4.28448e+09 -15

114. 16 4: -4.29497e+09 1.4354e+09 1.7527e+09 0 -2.10144e+08 -10 -1
 1.7155e+09

115. 17 4: 8.77875e+08 2.18228e+08 3.1838e+09 5.72301e+09 0 -3 0
 1.63835e+07

116. 18 4: -5 4.01726e+09 0 1.08241e+09 8.77875e+08 0 0 -2.7797e+09

117. 19 4: -9 -2.14748e+09 0 0 -5 1.35301e+09 -6 0

118. 20 4: -26 0 0 0 -6 0 -33 0

119. 21 4: -39 0 0 0 -34 0 0 0

120. 22 4: 0 0 0 -5 -30 0 -10 -5

121. 0 5: -6.47321e+10 -4.94733e+10 -250 -5.167e+10 -255 -260
 -8.062e+09 -4.31198e+10

122. 1 5: -4.12351e+08 -1.71134e+10 -8.73122e+09 -1.71326e+10
 -4.36613e+10 -1.22784e+10 -1.41996e+10 3.48147e+09

123. 2 5: -3.65632e+09 3.66775e+08 -3.43363e+10 -1.84662e+10
 -6.80155e+09 -4.25247e+10 -2.31641e+10 -4.26302e+10

124. 3 5: -1.68005e+10 -2.29137e+10 -1.56502e+10 -8.94076e+08
 -6.51303e+09 -1.83365e+09 -2.22898e+10 -285

125. 4 5: -6.61337e+10 -350 -4.81038e+10 -3.77912e+10 -2.84421e+10
 -350 -6.3296e+10 -350

126. 5 5: -3.97365e+10 -105 -7.04562e+08 2.68138e+08 -9.71722e+09 -115
 -2.56403e+10 -95

127. 6 5: -7.89232e+10 -355 -2.04488e+10 -8.27027e+10 -8.40225e+10
 -355 -8.21573e+10 -360

128. 7 5: -3.98581e+10 -70 4.27072e+09 1.56358e+10 -1.88569e+10 -60
 -3.89475e+10 -75

129. 8 5: -2.70478e+10 -4.50965e+10 -3.27574e+10 -3.65059e+10
 -2.942e+10 -100 2.713e+08 -1.69964e+10

130. 9 5: -64000 9.00776e+07 1.62182e+10 -8.38317e+09 -1.44858e+10
 -2.57406e+10 -1.78656e+10 -2.57698e+10

131. 10 5: -1.88901e+09 -2.85597e+10 -1.47254e+10 -2.57698e+10
 -4.29502e+09 -1.46967e+10 -1.28849e+10 -1.33057e+10

132. 11 5: -3.92452e+09 -6.58273e+09 -1.28849e+10 4.94026e+09
 2.48551e+09 -6.44245e+09 -4.95593e+09 6.74332e+08

133. 12 5: -2.57563e+10 -1.5025e+10 -2.06767e+10 -1.61768e+10
 -2.78957e+10 -5.31458e+08 -1.99872e+10 -1.71942e+10

134. 13 5: -1.35448e+10 -1.28849e+10 5.98427e+09 -1.71479e+10
 -2.08108e+10 -1.42396e+09 8.52034e+08 1.08727e+09

135. 14 5: 1.34822e+10 -1.38028e+09 -4.57564e+09 -35 -6.32011e+09
 -4.26063e+09 -15 -35

136. 15 5: 0 0 0 0 0 0 0 0

137. 16 5: -8 -2 -15 2.18664e+08 -5 0 0 -3.53231e+08

138. 17 5: -3.31995e+09 -1.2509e+09 1.74931e+08 0 1.38474e+09
 2.72785e+08 6.55342e+06 0

139. 18 5: 1.65349e+09 1.02397e+07 7.17263e+08 0 -1.89592e+08
 -1.53024e+09 8.11809e+08 8.45627e+08

140. 19 5: 0 2.23244e+09 0 8.73778e+08 1.11229e+09 3.3772e+09 0
 2.27546e+08

141. 20 5: 0 -5 -2.7797e+09 0 0 0 0 8.19167e+08

142. 21 5: 0 -36 0 0 0 2.90931e+09 0 0
 143. 22 5: 0 0 1.55163e+09 -10 0 -63 -10 -35
 144. 0 6: 9.6177e+09 -7.22528e+09 -105 -1.03716e+10 -135 -90
 -2.19695e+10 -2.73852e+10
 145. 1 6: -2.35563e+10 -2.97942e+10 -1.61878e+10 5.28705e+08
 2.82547e+09 -9.31538e+09 -2.61152e+10 -2.53086e+10
 146. 2 6: -1.35886e+10 -5.17037e+10 -4.89855e+10 -2.39666e+10
 -2.89169e+10 -5.58346e+10 -2.91628e+10 -3.66001e+10
 147. 3 6: -8.21968e+09 -8.21176e+09 1.37749e+09 -85 3.77922e+09
 -5.03645e+10 -75 -90
 148. 4 6: 0 0 0 0 0 0 0 0 0
 149. 5 6: 0 0 0 0 0 0 0 0 0
 150. 6 6: 0 0 0 0 0 0 0 0 0
 151. 7 6: 0 0 0 0 0 0 0 0 0
 152. 8 6: -2.19594e+09 6.95981e+09 -50 -8.70982e+09 -60 -45
 -1.92811e+10 -3.1131e+10
 153. 9 6: -2.96819e+10 -1.09846e+10 -2.50842e+10 -3.01562e+10
 1.51069e+10 -4.63346e+08 -3.26706e+09 -1.72549e+10
 154. 10 6: -1.57613e+09 3.94402e+08 -5.50175e+09 -2.56831e+09
 -1.28822e+10 -1.50324e+10 -1.80217e+10 2.81164e+09
 155. 11 6: -1.07374e+10 3.43361e+08 -1.07374e+10 -1.19663e+10
 -4.51768e+09 -7.70671e+09 2.50898e+09 -2.67138e+09
 156. 12 6: 3.0706e+09 9.5355e+09 -2.1384e+10 -1.28397e+10 -1.50324e+10
 -6.04127e+09 -7.58744e+09 -3.08312e+09
 157. 13 6: -9.94558e+09 -3.7879e+09 -5.25859e+09 9.57043e+09
 4.13439e+09 -3.91474e+09 -8.30843e+09 -4.23255e+09
 158. 14 6: -1.22579e+10 -1.45747e+10 -1.15613e+10 -45 -1.49148e+10
 -1.18509e+10 -45 -35
 159. 15 6: 0 0 0 0 0 0 0 0 0
 160. 16 6: 348423 -1.80854e+09 -15 2.6869e+07 -15 -5 7.17263e+08
 1.36112e+06

161. 17 6: 1.30666e+06 -4.29497e+09 2.14952e+07 -5.32173e+09 -6
 -2.14748e+09 3.80913e+09 -4.26138e+09

162. 18 6: 1.56631e+09 -1.09046e+09 3.5967e+09 2.7794e+09 4.36456e+08
 -7.34479e+08 -5.34507e+09 1.05176e+07

163. 19 6: -4.29497e+09 5.27228e+06 0 -12 0 1.14549e+08 1.21221e+09
 -10

164. 20 6: 3.63664e+09 -20 -6 -16 -1.86305e+09 -10 0 -15

165. 21 6: 0 -20 -1.80616e+09 0 0 -25 0 -10

166. 22 6: 0 -20 3.64074e+08 -35 0 -10 -30 -10

167. 0 7: -2.72701e+10 -8.31755e+09 -120 -1.55213e+10 -115 -130
 -7.07415e+09 -1.35365e+10

168. 1 7: -1.63055e+10 -5.2061e+09 -1.55105e+09 -1.97152e+10
 -3.4323e+10 -1.16769e+10 4.39187e+09 -1.02996e+10

169. 2 7: 8.67656e+09 -2.19842e+09 -2.38508e+10 -2.25433e+10
 -1.32084e+10 -1.90085e+09 5.70776e+09 -3.43597e+10

170. 3 7: -1.53869e+10 -3.95682e+10 -3.20342e+10 -140 2.99219e+09
 -1.48801e+10 -75 -115

171. 4 7: 0 0 0 0 0 0 0 0 0

172. 5 7: 0 0 0 0 0 0 0 0 0

173. 6 7: 0 0 0 0 0 0 0 0 0

174. 7 7: 0 0 0 0 0 0 0 0 0

175. 8 7: -4.50955e+10 -1.92524e+10 -160 -4.32068e+10 -165 -170
 -2.73499e+10 -5.30944e+09

176. 9 7: -4.48862e+09 -1.21985e+10 3.87172e+09 3.56353e+08 -2.1042e+10
 -6.81574e+08 -1.58469e+10 -9.66368e+09

177. 10 7: -1.22734e+10 3.78372e+09 -1.53071e+10 -3.33595e+09
 -2.55449e+09 -5.10993e+09 -6.72302e+09 -9.75458e+08

178. 11 7: -1.438e+09 -1.70408e+09 -4.71132e+09 -6.87675e+07
 -2.14727e+10 -2.73769e+09 -7.96363e+09 -2.31704e+10

179. 12 7: -5.91792e+09 -1.77491e+10 2.58734e+09 -9.70874e+09
 -6.71675e+09 -2.99536e+09 -4.91638e+09 -1.14085e+10

180. 13 7: -6.30892e+09 -2.14736e+10 1.56221e+09 -8.17984e+09
 -2.14748e+09 -2.14748e+09 3.84083e+09 -3.56141e+09

181. 14 7: 3.16964e+09 1.20372e+09 -1.5238e+09 -25 -1.10261e+10
 -1.07374e+10 -20 -50

182. 15 7: 0 0 0 0 0 0 0 0

183. 16 7: 4.08279e+08 2.15529e+08 -15 6.77174e+09 -10 -45 5.3738e+07
 -2.04236e+09

184. 17 7: 2.82013e+09 3.54245e+09 -7.33925e+09 -5.70845e+09
 2.74254e+08 -3.09866e+09 1.02397e+07 -2.33681e+09

185. 18 7: 8.65037e+08 4.716e+07 1.21834e+10 2.43114e+08 -1.03372e+09
 -2.90483e+09 -18 2.36875e+09

186. 19 7: -9 1.70145e+09 -4.28473e+09 0 0 -3.99108e+09 -15 -20

187. 20 7: 0 0 0 0 0 0 0 0

188. 21 7: 0 0 0 0 0 0 0 0

189. 22 7: 0 0 0 -5 0 0 -5 -5

190. 0 8: -3.89851e+09 -3.18093e+09 -55 8.03998e+08 -70 -135
 -1.90478e+10 -9.73317e+08

191. 1 8: -2.30222e+10 -9.46482e+09 -1.00503e+10 -2.85048e+09
 -8.07219e+09 -3.65302e+09 -1.657e+10 -5.50749e+09

192. 2 8: -2.49192e+10 -1.47781e+10 -2.82328e+09 -2.77702e+10
 -1.60097e+10 -1.45672e+10 -1.34307e+10 -1.84942e+10

193. 3 8: -5.67817e+09 -8.58993e+09 -8.54859e+09 -1.05536e+10
 -2.37126e+10 -5.30854e+09 -35 7.45615e+09

194. 4 8: -20 1.03859e+10 -1.50324e+10 -2.2796e+09 -1.1691e+10
 -1.28849e+10 -55 -2.35051e+09

195. 5 8: -45 -1.07374e+10 -1.68081e+10 -1.28849e+10 -75 4.7516e+09
 -60 -5.31632e+09

196. 6 8: -10 6.41247e+08 -6.44245e+09 -1.07374e+10 -5 -4.77456e+09
 -30 6.95355e+09

197. 7 8: -25 1.08661e+09 -3.99419e+09 -1.98147e+08 -25 -7.94871e+09
 -713 4.28591e+09

198. 8 8: 2.24546e+06 -3.571e+09 -7.20073e+09 -5.15412e+09 -40
 2.43665e+09 -7.08999e+09 -6.43567e+09

199. 9 8: -5.37533e+09 -8.63366e+09 -8.05412e+09 -1.16398e+10
 8.54732e+08 -1.29374e+09 -8.61498e+09 -3.84541e+09

200. 10 8: -1.55813e+09 -1.91873e+09 -1.35594e+09 -2.88149e+08
 -7.00751e+09 -1.57869e+09 1.17211e+06 -6.50832e+09

201. 11 8: -1.39223e+09 -3.57564e+09 -1.78359e+08 -5.22779e+09
 -9.04841e+09 -5.01207e+09 -2.06886e+09 -6.44183e+09

202. 12 8: 1.2054e+10 -3.18574e+09 7.59145e+08 -6.44245e+09 7.48109e+08
 -8.60081e+09 -3.67446e+09 -4.77184e+09

203. 13 8: 8.03175e+08 -2.14527e+09 -3.34918e+09 3.04589e+09
 8.66826e+09 -1.5884e+09 2.23194e+09 3.77624e+09

204. 14 8: -2.04317e+09 1.26911e+09 -1.28849e+10 -5.72935e+09
 6.90614e+08 409720 -35 -5.31778e+09

205. 15 8: -15 -1.62253e+10 -2.13257e+09 -4.22105e+09 2.88234e+09
 632552 -1.08126e+09 -5.42284e+09

206. 16 8: -6.19082e+09 28231 -4.70979e+08 -4.65116e+09 -45
 -4.29497e+09 1.46322e+09 -1.9325e+08

207. 17 8: -8.1571e+08 5.46799e+09 -2.46344e+08 -1.8464e+09
 -2.14748e+09 -1.87231e+09 -1.80389e+09 3.77902e+08

208. 18 8: -5.38543e+09 6.50876e+09 -2.63236e+09 2.87683e+09
 -2.68798e+09 1.17936e+09 2.65616e+08 -3.92419e+09

209. 19 8: 1.87275e+07 6.2683e+09 -4.29497e+09 -25 -6.07545e+09
 -6.74526e+08 -20 0

210. 20 8: 0 0 0 0 0 0 0 0

211. 21 8: 0 0 0 0 0 0 0 0

212. 22 8: 0 0 0 -5 0 0 -5 -5

213. 0 9: -1.15985e+10 6.27111e+08 -45 -1.13145e+10 -30 -50
 6.98045e+08 -7.86259e+09

214. 1 9: -6.55181e+09 -1.40826e+10 -5.28379e+09 -4.31972e+09
 -1.43892e+10 -3.29031e+08 7.02383e+09 -1.02095e+10

215. 2 9: -3.21522e+09 -1.0024e+09 -3.22446e+09 -3.65448e+10
 -2.17426e+09 -1.40946e+10 -1.50303e+10 -1.71402e+10

216. 3 9: -2.14748e+10 -7.49638e+09 -8.71203e+08 7.31934e+09
 -5.50631e+09 -3.54659e+09 -1.06786e+10 -8.61398e+09

217. 4 9: -3.22123e+10 -1.2752e+10 -2.49792e+10 -3.43597e+10
 -1.93274e+10 -2.64459e+10 -1.49893e+10 -1.76187e+10

218. 5 9: -7.79349e+09 -4.79607e+09 1.43537e+10 2.25066e+09
 -4.73612e+09 -2.25603e+09 -1.13293e+10 -4.11142e+09

219. 6 9: -9.10524e+09 -5.17789e+09 -1.71758e+10 8.01535e+08
 -6.02774e+08 -6.44245e+09 -2.57698e+10 -1.07374e+10

220. 7 9: -1.28828e+10 -2.38817e+10 -1.35961e+10 -3.83742e+09
 -1.7121e+10 -1.03805e+10 -3.95725e+09 -2.57698e+10

221. 8 9: -1.77716e+10 -1.71799e+10 6.78517e+09 -1.13913e+10
 -1.0251e+10 -2.59483e+10 -8.37063e+09 -3.56299e+09

222. 9 9: -6.36047e+09 -1.07374e+10 1.5527e+08 -4.35208e+09
 -1.24178e+10 -1.24936e+10 -4.47429e+09 2.69674e+09

223. 10 9: -2.14748e+09 8.96088e+08 2.36833e+07 -5.08137e+09
 2.19366e+08 -4.29206e+09 3.00365e+09 -5.93232e+09

224. 11 9: 2.0055e+08 -2.14748e+09 -6.44223e+09 -1.47596e+09
 6.52064e+08 8.99161e+08 -4.7411e+09 -8.46176e+08

225. 12 9: -1.7173e+09 -1.46419e+09 -2.92929e+09 2.9462e+09 761958
 -3.90179e+09 -5.14728e+09 -4.29497e+09

226. 13 9: -1.07374e+10 -1.50308e+10 -2.01758e+08 1.38354e+06
 -3.42134e+09 -1.42584e+08 -6.43567e+09 -3.44417e+09

227. 14 9: -2.82756e+09 -3.69107e+09 -2.14636e+09 -1.40226e+10
 -1.58847e+10 -1.49714e+10 -6.79096e+09 -3.21205e+10

228. 15 9: -6.12595e+09 -1.0218e+09 1.85899e+09 -2.113e+09 2.47874e+09
 1.20929e+09 -3.75593e+09 6.16708e+09

229. 16 9: 6.59183e+09 -6.67527e+08 -1.28466e+10 -6.81e+09 -1.68839e+09
 -1.67569e+10 -8.74621e+09 -6.39318e+09

230. 17 9: -1.03329e+10 1.22048e+09 -1.12447e+09 -4.41483e+09 -11
 1.47376e+09 -1.88172e+09 -4.29493e+09

231. 18 9: -8.27403e+09 2.53453e+09 -8.13864e+09 7.05577e+06
 -3.96454e+08 -2.69083e+09 1.60976e+09 -9.85037e+07

232. 19 9: -9.571e+08 5.32669e+08 -1.68456e+09 -25 -3.44623e+09
 -4.29497e+09 -30 -25

233. 20 9: 0 0 0 0 0 0 0 0
 234. 21 9: 0 0 0 0 0 0 0 0
 235. 22 9: 0 0 0 -5 0 0 -5 -5
 236. 0 10: -1.3679e+10 -80 -90 -1.37193e+10 -75 -75 -4.70928e+09 -75
 237. 1 10: 5.98622e+08 -40 2.66195e+09 -1.00252e+10 -1.28432e+10 -55
 -9.76036e+09 -45
 238. 2 10: -1.95661e+10 -65 -1.42617e+10 -8.94979e+09 -5.50698e+09 -70
 -1.71799e+10 -65
 239. 3 10: -6.43835e+09 -3.81524e+09 -4.29497e+09 -2.2163e+09
 2.03409e+09 -35 3.22561e+09 -8.49285e+09
 240. 4 10: 4.18998e+09 -1.18631e+10 -7.40447e+09 -9.67326e+09
 -1.50324e+10 -9.35793e+09 -2.14013e+10 -9.50733e+09
 241. 5 10: -1.09363e+10 -3.34094e+09 -8.5087e+09 2.4939e+09 3.00143e+09
 -6.56228e+09 -3.56661e+09 1.80877e+10
 242. 6 10: 4.9642e+09 2.26348e+09 -1.77685e+10 -2.14748e+09
 -1.76516e+09 -7.10366e+09 -1.1221e+09 -3.58244e+09
 243. 7 10: 1.11748e+09 -2.44068e+09 -6.01216e+09 -1.22862e+10
 1.80106e+09 7.7528e+09 2.08517e+08 -5.6539e+09
 244. 8 10: -4.65806e+09 -8.30325e+08 -1.07374e+10 -3.09027e+09
 5.73249e+09 -2.22816e+09 -1.07306e+10 1.79992e+09
 245. 9 10: -4.99503e+09 1.99024e+09 -2.80233e+09 2.09112e+09
 -1.80561e+09 -2.58738e+09 -1.72474e+09 -40
 246. 10 10: -1.50324e+10 -20 -9.46445e+08 -4.89807e+09 -1.02737e+09
 -20 -6.43861e+09 -40
 247. 11 10: -6.44125e+09 -25 8.9839e+08 1.31749e+09 -2.55477e+09 -20
 -5.40169e+09 -40
 248. 12 10: -1.38897e+08 -30 -1.20484e+10 -1.50324e+10 -7.64915e+09
 -25 -7.57392e+09 -30
 249. 13 10: -8.16549e+09 -6.44245e+09 -7.26316e+06 3.73527e+09
 5.46634e+08 -40 4.37436e+09 2.75201e+09
 250. 14 10: -8736 1.24525e+09 -1.16853e+07 -1.28729e+10 -4.28181e+09
 -9.68967e+08 -1.07374e+10 -6.07793e+09

251. 15 10: -1.10109e+10 -2.10001e+09 -1.09273e+09 1.98167e+10
 4.89438e+07 5.56185e+08 -4.9338e+09 -7.86287e+09

252. 16 10: -4.60024e+09 -4.88e+08 -9.46377e+09 -8.24678e+09
 -4.91636e+09 -3.58552e+09 -2.52796e+09 -1.62368e+09

253. 17 10: 2.01909e+09 2.00229e+09 3.46644e+09 -1.75832e+09
 -8.21185e+08 3.11124e+09 -1.01648e+09 -2.14743e+09

254. 18 10: 2.0053e+09 -3.6851e+09 -6.31761e+09 5.7489e+09 0
 -3.10078e+09 3.31041e+06 1.16948e+06

255. 19 10: 6.60664e+08 -4.34702e+08 -7.08801e+07 5.84427e+06
 9.49171e+07 -6.44245e+09 -30 1.26249e+09

256. 20 10: -10 9.21125e+08 5.2589e+08 462 3497 10862 -15
 9.77022e+08

257. 21 10: -15 9.5443e+06 7.484e+08 0 -20 0 -20 0

258. 22 10: 0 0 0 -5 0 127029 -5 -5

259. 0 11: 0 0 -5 0 -5 -5 0 0

260. 1 11: 0 0 0 0 0 0 0 0

261. 2 11: 0 0 0 0 0 0 0 0

262. 3 11: -1.60863e+10 2.42598e+09 -55 -1.91591e+09 -10 -25
 7.19523e+08 2.48158e+09

263. 4 11: -4.27858e+09 -3.99324e+09 -6.83033e+09 -1.06283e+10
 -1.28849e+10 -2.14092e+09 -6.33691e+08 -2.14748e+09

264. 5 11: -9.95535e+09 -5.99743e+09 -6.44246e+09 1.00618e+10
 -9.65726e+09 -3.06575e+09 -8.71358e+09 8.16286e+08

265. 6 11: -8.92663e+09 -1.08592e+10 -6.36208e+09 1.86369e+09
 -1.71799e+10 4.38177e+08 -2.0487e+10 -1.24597e+10

266. 7 11: -7.87171e+09 -4.73249e+09 1.47102e+08 -6.44245e+09
 -5.21566e+09 -9.86757e+09 -4.49517e+09 -6.26349e+09

267. 8 11: -6.45818e+09 -4.55621e+09 -3.06854e+09 7.89112e+09
 -3.03145e+09 -6.65091e+09 -8.73775e+09 -2.34512e+09

268. 9 11: -7.39162e+09 -9.73116e+09 -8.48925e+09 -45 -8.58821e+09
 -3.76497e+06 -45 -50

269. 10 11: 0 0 0 0 0 0 0 0

270. 11 11: 0 0 0 0 0 0 0 0 0
 271. 12 11: 0 0 0 0 0 0 0 0 0
 272. 13 11: -1.50324e+10 -1.48869e+09 -15 -2.87398e+09 -25 -25
 -6.37906e+09 -9.67879e+08
 273. 14 11: -2.30255e+09 -6.74716e+09 -5.33339e+09 -4.19364e+09
 -1.28838e+10 -1.97271e+09 -1.07361e+10 -2.94492e+09
 274. 15 11: -8.57719e+09 -4.29011e+09 -5.76593e+09 -2.77791e+09
 8.42249e+08 3.11746e+07 -2.92728e+09 -4.86787e+09
 275. 16 11: -2.43748e+09 -3.81452e+09 -5.69605e+07 2.70879e+09
 -1.2922e+10 -5.60498e+09 -1.05953e+09 -1.9095e+09
 276. 17 11: -2.14748e+09 -1.01879e+08 1.5199e+09 -4.29215e+09 -388
 -2.1192e+09 -1.18352e+09 1.43322e+09
 277. 18 11: 0 1.1181e+09 2.5733e+08 6.16396e+08 -2.14748e+09
 2.04148e+09 -1.99562e+09 5.73516e+09
 278. 19 11: 7.2739e+08 2.86751e+09 -4.29356e+09 3.0301e+07 -5.68912e+09
 31612 328147 1.13609e+09
 279. 20 11: 5.81615e+06 1.5781e+09 0 -2.53999e+08 1.37434e+09
 1.04139e+06 7.95396e+06 1.50891e+09
 280. 21 11: 9.20044e+06 1.21746e+09 3.02959e+09 0 7.82266e+08
 1.31125e+08 0 0
 281. 22 11: 0 0 0 -10 0 -2.14512e+09 -15 -20
 282. 0 12: 0 0 -5 0 -5 -5 0 0
 283. 1 12: 0 0 0 0 0 0 0 0 0
 284. 2 12: 0 0 0 0 0 0 0 0 0
 285. 3 12: -3.49428e+06 3.24166e+09 -25 -1.07125e+10 -35 -5
 -1.07218e+10 -1.58307e+09
 286. 4 12: -6.43222e+09 -7.93145e+08 -3.01596e+09 -8.32426e+09
 -7.2526e+09 4.33995e+09 -4.29497e+09 -2.14748e+09
 287. 5 12: -9.17399e+08 -2.21142e+09 -2.29298e+09 -4.29497e+09
 -1.59122e+09 -3.32861e+09 -1.31305e+09 0
 288. 6 12: -3.31386e+09 4.49149e+09 -8.9359e+09 -6.78025e+09
 -1.04247e+10 -3.97567e+09 -8.09392e+09 3.39994e+09

289. 7 12: 4.35163e+09 5.25231e+08 -6.25266e+09 -3.20404e+09
 1.67931e+09 -3.04697e+09 -1.07374e+10 -1.48871e+09

290. 8 12: -6.36256e+09 2.18019e+09 -4.14744e+09 -3.64238e+09
 -2.75057e+09 0 2.88268e+09 1.43776e+09

291. 9 12: 3.51049e+09 -7.13307e+08 -1.00644e+09 -35 -1.2087e+10
 1.85579e+09 -30 -15

292. 10 12: 0 0 0 0 0 0 0 0

293. 11 12: 0 0 0 0 0 0 0 0

294. 12 12: 0 0 0 0 0 0 0 0

295. 13 12: -3.12918e+09 2.79655e+08 -40 -4.85553e+08 -30 -35
 -1.81723e+09 -40

296. 14 12: 2.72691e+09 -45 -4.29698e+09 -5.68424e+09 -5.49877e+09 -40
 -1.37727e+10 -50

297. 15 12: -3.24813e+09 -30 -2.14748e+09 -4.29496e+09 -2.14749e+09
 -40 3.3509e+09 -20

298. 16 12: -2.14063e+09 -55 -2.14748e+09 1.0526e+09 -3.90348e+09 -35
 2.80817e+08 -70

299. 17 12: -1.00091e+09 1.64361e+09 -4.05748e+09 0 -2.80329e+09 -50
 -4.87492e+09 1.06563e+10

300. 18 12: -4.27045e+09 1.06065e+09 5577 3.62744e+09 -2.14212e+09
 40308 2.77077e+06 -1.60064e+09

301. 19 12: -2.14078e+09 -2.14551e+09 -2.5242e+09 -7.13121e+09
 -8.58993e+09 -2.14735e+09 1.81362e+09 -2.14545e+09

302. 20 12: -2.14747e+09 1.13524e+09 2.86768e+09 2.44858e+09
 9.20049e+08 -5.30886e+09 6.06388e+09 2152

303. 21 12: 7763 4224 -2.14727e+09 0 1.85508e+09 -2.14548e+09 0 0

304. 22 12: 0 0 4.84287e+08 -25 1.21184e+09 51884 -10 -10

305. 0 13: 0 0 -5 0 -5 -5 0 0

306. 1 13: 0 0 0 0 0 0 0 0

307. 2 13: 0 0 0 0 0 0 0 0

308. 3 13: -6.39092e+09 0 -20 -2.13533e+09 -35 -25 -4.84709e+09 0

309. 4 13: -2.14748e+09 0 -9.39536e+08 -8.58993e+09 -1.35801e+09 0
 -1.69484e+09 0

310. 5 13: -2.14748e+09 0 0 6.67099e+09 -4.35096e+09 0 -1.07285e+10
 0

311. 6 13: -2.99322e+09 0 -1.57282e+09 1.6944e+09 1.0557e+09 0
 -3.37587e+09 0

312. 7 13: -9.47686e+09 0 -1.58393e+09 -1.11732e+09 -1.07669e+10 0
 -4.29497e+09 0

313. 8 13: 5.99156e+08 0 1.05046e+09 -1.34524e+09 -9.13445e+09 0
 -2.84514e+09 0

314. 9 13: -4.68923e+09 0 -4.66743e+08 -10 -3.50285e+08 0 -40 0

315. 10 13: -10 0 0 -1 -3 0 -10 0

316. 11 13: -5 0 -2 -4 -5 0 0 0

317. 12 13: -5 0 -11 -5 -10 0 -4.29497e+09 0

318. 13 13: 3.00865e+09 0 -29 -10 -5 0 -5 -15

319. 14 13: 0 0 0 0 0 0 0 0

320. 15 13: 0 0 0 0 0 0 0 0

321. 16 13: 0 0 0 0 0 0 0 0

322. 17 13: -2.14747e+09 -15 -20 2.05451e+09 -10 -15 -6.44243e+09
 -10

323. 18 13: -4.29495e+09 0 1.64361e+09 2.56813e+09 -2.21227e+09 -10
 5.74291e+09 0

324. 19 13: 4.25015e+09 0 -2.14736e+09 1.91427e+06 -1.68827e+09 0
 2.26913e+09 0

325. 20 13: 2.38853e+06 0 27993 4144 5.76931e+09 0 0 0

326. 21 13: 192 0 0 0 64455 0 0 500

327. 22 13: 0 300 160 -10 1.05584e+07 0 -25 -10

328. 0 14: 0 -5 -5 0 -5 -5 0 -5

329. 1 14: 0 -5 0 0 0 -5 0 -5

330. 2 14: 0 -5 0 0 0 -5 0 -5

```

331. 3 14: -3.38412e+08 -20 -5 0 -10 -20 2.30851e+09 -20
332. 4 14: -8.58431e+09 -35 0 0 1.45463e+09 -15 -1.91805e+08 -25
333. 5 14: -2.14748e+09 -25 0 0 -1.79149e+09 -40 3.24422e+09 -25
334. 6 14: 7.60324e+09 -30 0 0 1.91431e+08 -20 1.05904e+09 -20
335. 7 14: 1.6944e+09 -30 0 0 8.17766e+08 -20 -1.40165e+07 -45
336. 8 14: -7.56023e+08 -45 0 0 0 -35 -6 -45
337. 9 14: -5.01693e+09 -15 0 0 -2.13944e+09 -10 -13 -5
338. 10 14: -31 -10 0 0 -1 -5 -10 -10
339. 11 14: -4 -35 0 0 -22 -30 -52 -5
340. 12 14: -62 -35 0 0 -10 -15 -16 -15
341. 13 14: 2.04241e+09 -30 0 -35 -43 -25 -30 -40
342. 14 14: 0 -5 0 0 0 -5 0 -5
343. 15 14: 0 -5 0 0 0 -5 0 -5
344. 16 14: 0 -5 0 0 0 -5 0 -5
345. 17 14: 0 -5 0 0 0 -5 0 -5
346. 18 14: -2.13458e+09 -15 -10 0 -15 -20 -2.14735e+09 -5
347. 19 14: 132302 -10 0 0 1.11217e+09 -5 1.13337e+09 -10
348. 20 14: 0 -20 0 0 -2.1473e+09 -40 103288 -30
349. 21 14: 0 -5 0 700 0 -15 0 -10
350. 22 14: 0 0 0 0 0 -5 -5 0

```

▼

• 障碍物比例为30%时

共成功到达目标点13次，第一次成功规划路径是第3次训练，共耗费166步走到目标点，路径长度为164.894，但是后面的训练中成功到达目标点时，花费的步数都高于166，可见算法并未收敛，最后一次寻径成功是第163次，Q表状况与20%基本相同。

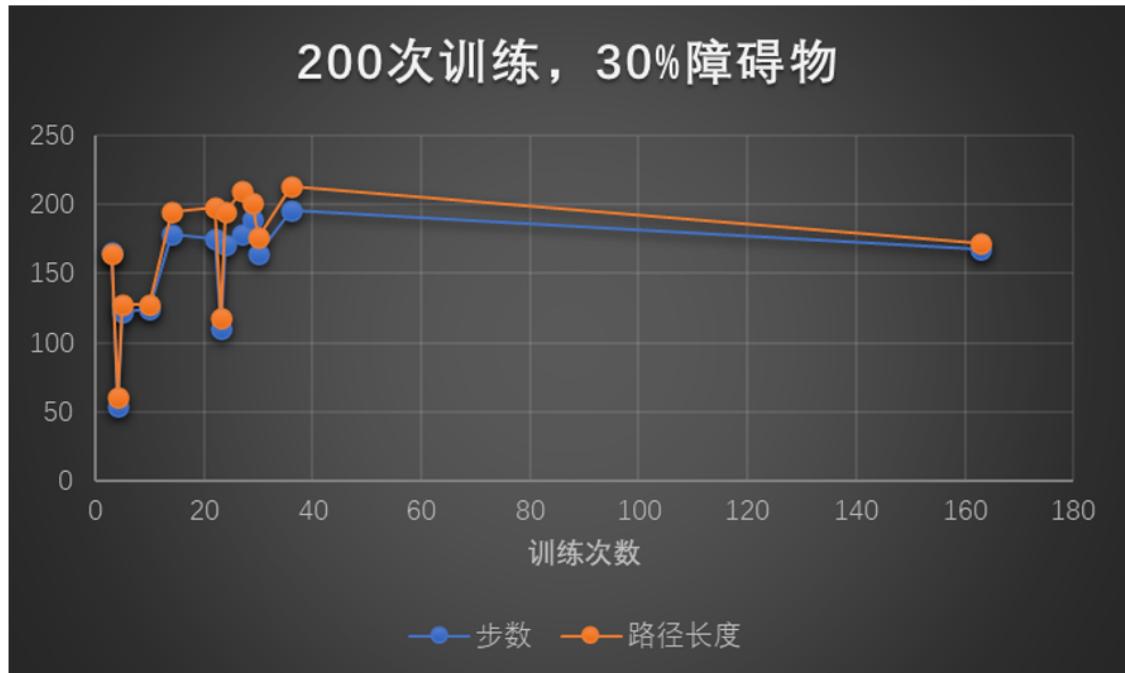


图 4.15 训练 200 次，障碍物比例为 30% 时的步数和路径长度随成功次数变化的图像

- 障碍物比例为50%时

共成功到达目标点7次，第一次成功规划路径是第13次训练，共耗费173步走到目标点，路径长度为 167.966，但是后面的训练中成功到达目标点时，花费的步数有的高于167.966，有的低于167.966，算法并未收敛，最后一次寻径成功是第163次，Q表状况与20%基本相同。

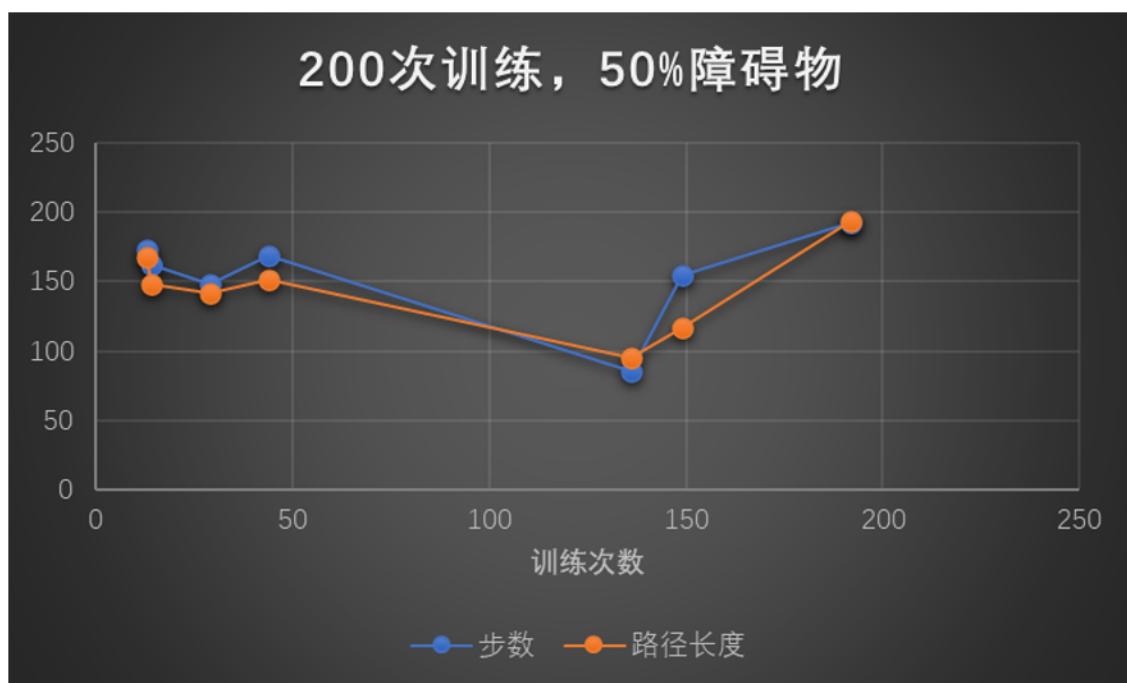


图 4.16 训练 200 次，障碍物比例为 50% 时的步数和路径长度随成功次数变化的图像

(2) 训练次数为1000次

- 障碍物比例为20%时

共成功到达目标点504次，第一次成功规划路径是第1次训练，共耗费101步走到目标点，路径长度为110.953，但是后面的训练中成功到达目标点时，花费的步数有的高于110.953，有的低于110.953，算法并未收敛，但是成功次数的比例约为百分之50，超过之前训练200次的，最后一次寻径成功是第998次，Q表状况与200次训练有所不同，到训练后期，已经出现了浮点数值超过可以表示的精度，显示为inf，如下：

```

998
Win
128.267

0 0: -1900 inf -1720 inf -1895 -1790 -1930 inf
1 0: -905 inf inf inf -875 inf -915 inf
2 0: -615 inf inf inf -660 inf -610 inf
3 0: -495 inf inf inf -510 inf -485 inf
4 0: -450 inf inf inf -525 inf -485 inf
5 0: -335 inf inf inf -335 inf -345 inf |
6 0: -225 inf inf inf -230 inf -260 inf
7 0: -190 inf inf inf -115 inf -175 inf
8 0: -105 inf inf inf -120 inf -80 inf
9 0: -125 inf inf inf -200 inf -220 inf
10 0: -150 inf inf -135 -100 inf -110 -135
11 0: -5 0 0 -5 0 -5 0
12 0: -5 0 0 -5 0 -5 0
13 0: -5 0 0 -5 0 -5 0
14 0: -10 inf -5 0 -5 0 -10 inf
15 0: -5 inf inf 0 -25 inf -10 inf
16 0: -10 inf inf inf -10 0 -10 inf
17 0: -30 inf inf -15 -20 inf -30 -20
18 0: -5 0 0 -5 0 -5 0
19 0: -5 0 0 -5 0 -5 0
20 0: -5 0 0 -5 0 -5 0
21 0: -5 0 0 -5 0 -5 0
22 0: -5 0 0 -5 -5 0 -5 -5
0 1: inf inf -920 inf -750 -940 inf inf
1 1: inf inf inf inf inf inf inf
2 1: inf inf inf inf inf inf inf
3 1: inf inf inf inf inf inf inf
4 1: inf inf inf inf inf inf inf
5 1: inf inf inf inf inf inf inf
6 1: inf inf inf inf inf inf inf
7 1: inf inf inf inf inf inf inf
8 1: inf inf inf inf inf inf inf
9 1: inf inf inf inf inf inf inf
10 1: inf inf inf -180 inf inf -170 -180

```

图 4.17 Q 表已经出现超出表示范围的值

行走步数与路径长度随成功次数的变化情况如下：

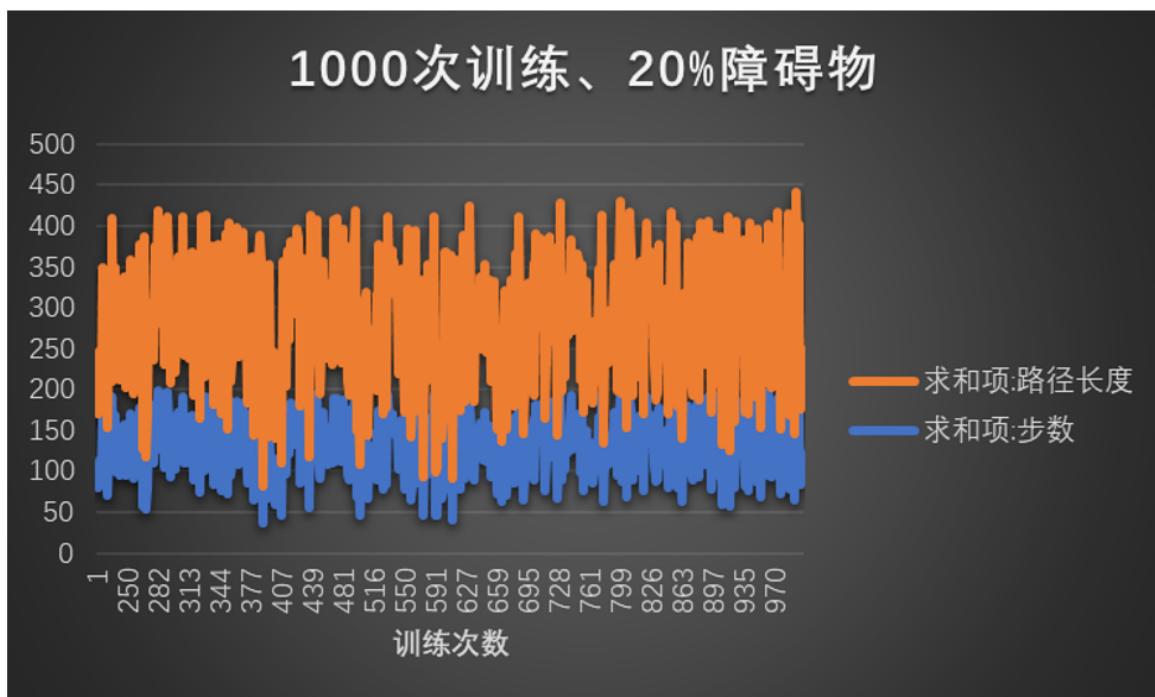


图 4.18 训练 1000 次，障碍物比例为 20% 时的步数和路径长度随成功次数变化的图像

根据图像可以看出路径长度与步数基本变化一致，且一直处于震荡状态，并未收敛。

- **障碍物比例为30%时**

共成功到达目标点450次，第一次成功规划路径是第3次训练，共耗费186步走到目标点，路径长度为184.693，但是后面的训练中成功到达目标点时，花费的步数有的高于184.693，有的低于184.693，算法并未收敛，但是成功次数的比例将近百分之50，比之前训练200次的高，最后一次寻径成功是第1000次，Q表状况与障碍物20%的相近。

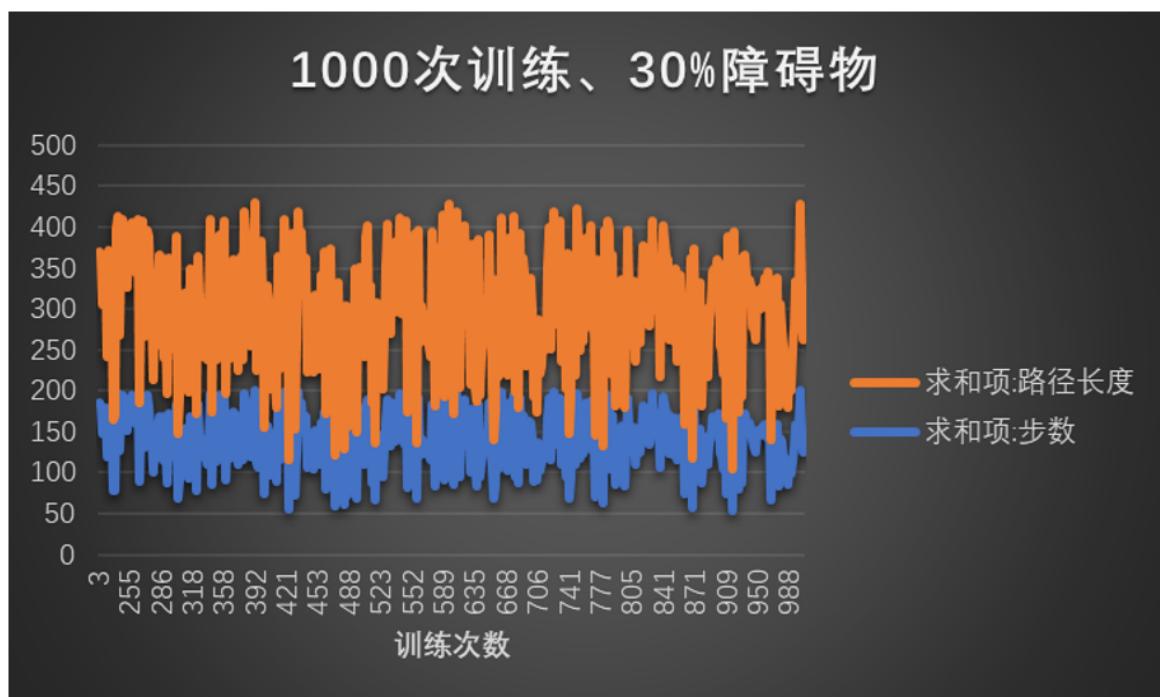


图 4.19 训练 1000 次，障碍物比例为 30% 时的步数和路径长度随成功次数变化的图像

- **障碍物比例为50%时**

共成功到达目标点19次，第一次成功规划路径是第1次训练，共耗费131步走到目标点，路径长度为110.468，但是后面的训练中成功到达目标点时，花费的步数有的高于110.468，有的低于110.468，算法并未收敛，成功次数极低，最后一次寻径成功是第895次，Q表状况与障碍物20%的相近。

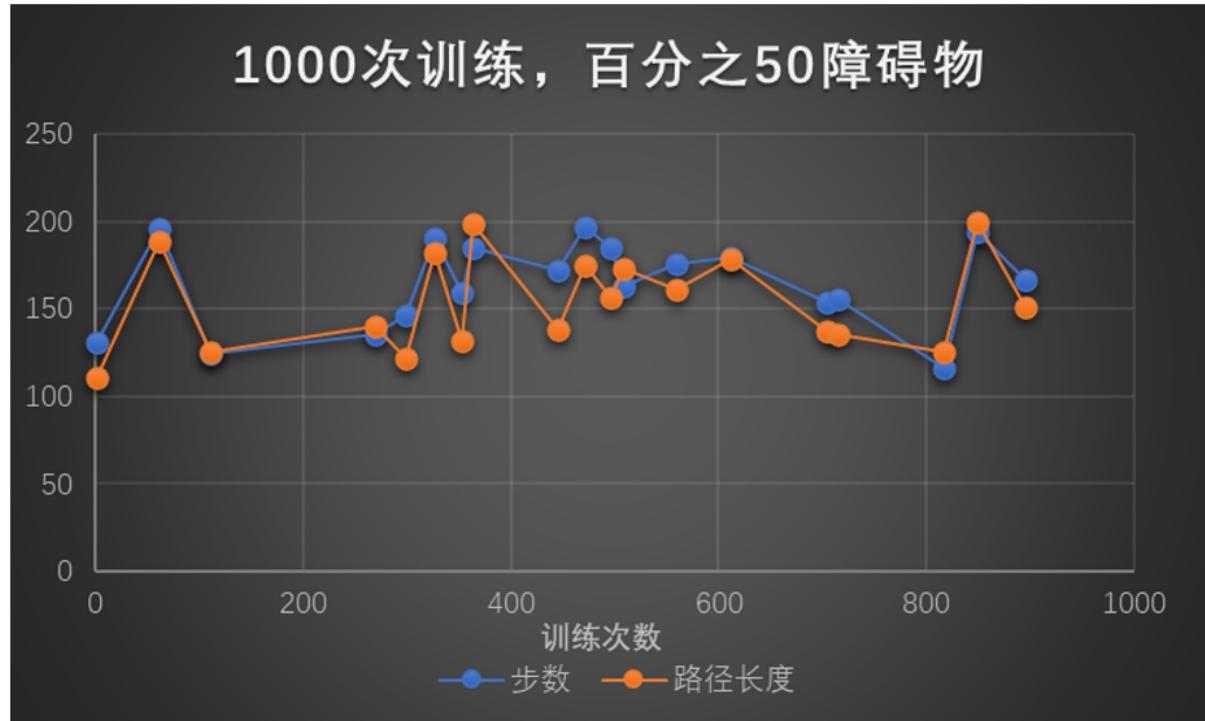


图 4. 20 训练 1000 次，障碍物比例为 50% 时的步数和路径长度随成功次数变化的图像
CSDN @_坐着云起时_

(3) 训练次数为5000次

到5000次训练时，算法仍然没有收敛，行走步数与路径长度随成功次数的变化情况如下：

障碍物比例为20%时，成功2789次，约占56%；障碍物比例为30%时，成功2655次，约占53%；障碍物比例为50%时，成功2182次，约占44%。

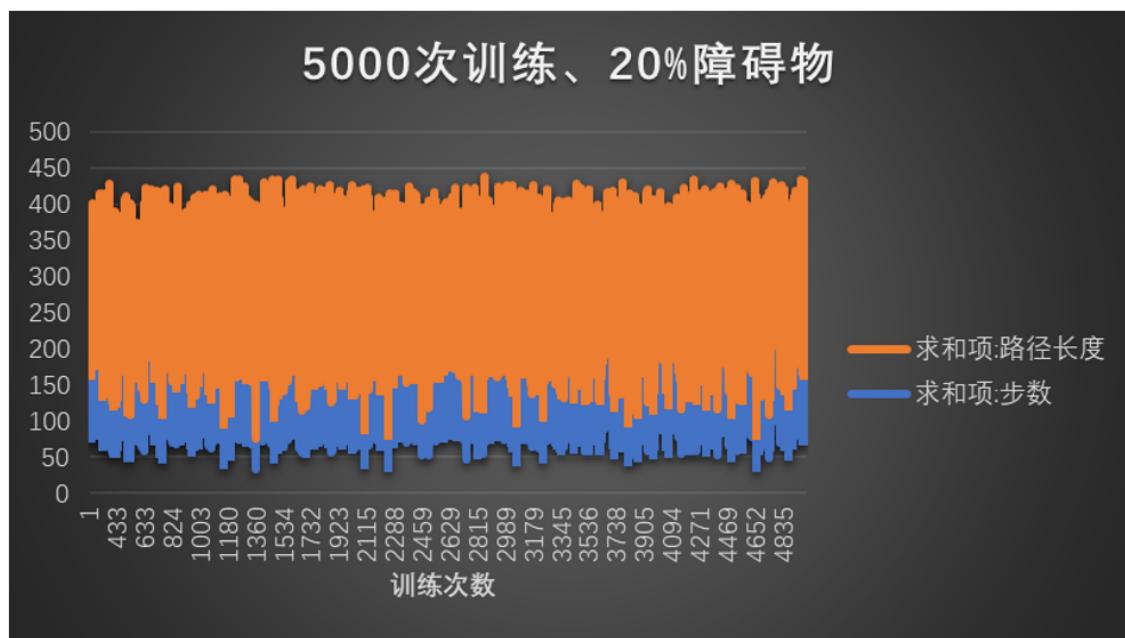


图 4. 21 训练 5000 次，障碍物比例为 20% 时的步数和路径长度随成功次数变化的图像
CSDN @_坐着云起时_

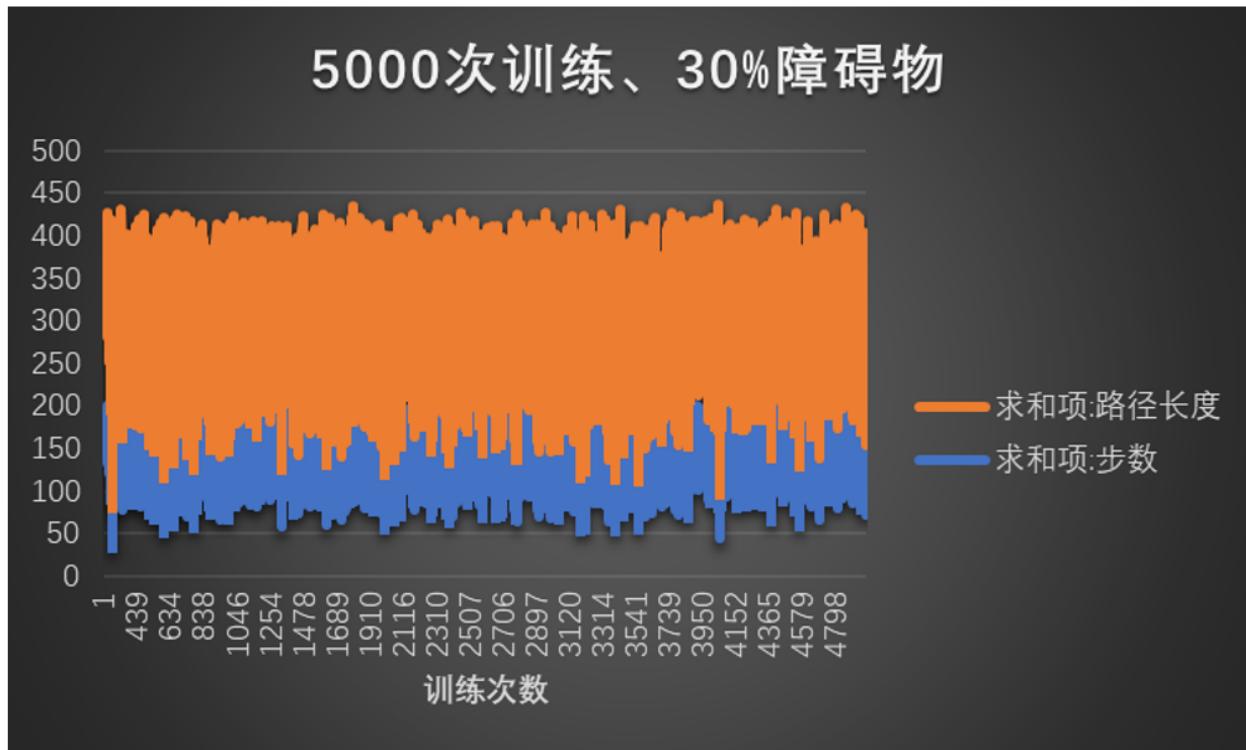


图 4.22 训练 5000 次，障碍物比例为 30% 时的步数和路径长度随成功次数变化的图像

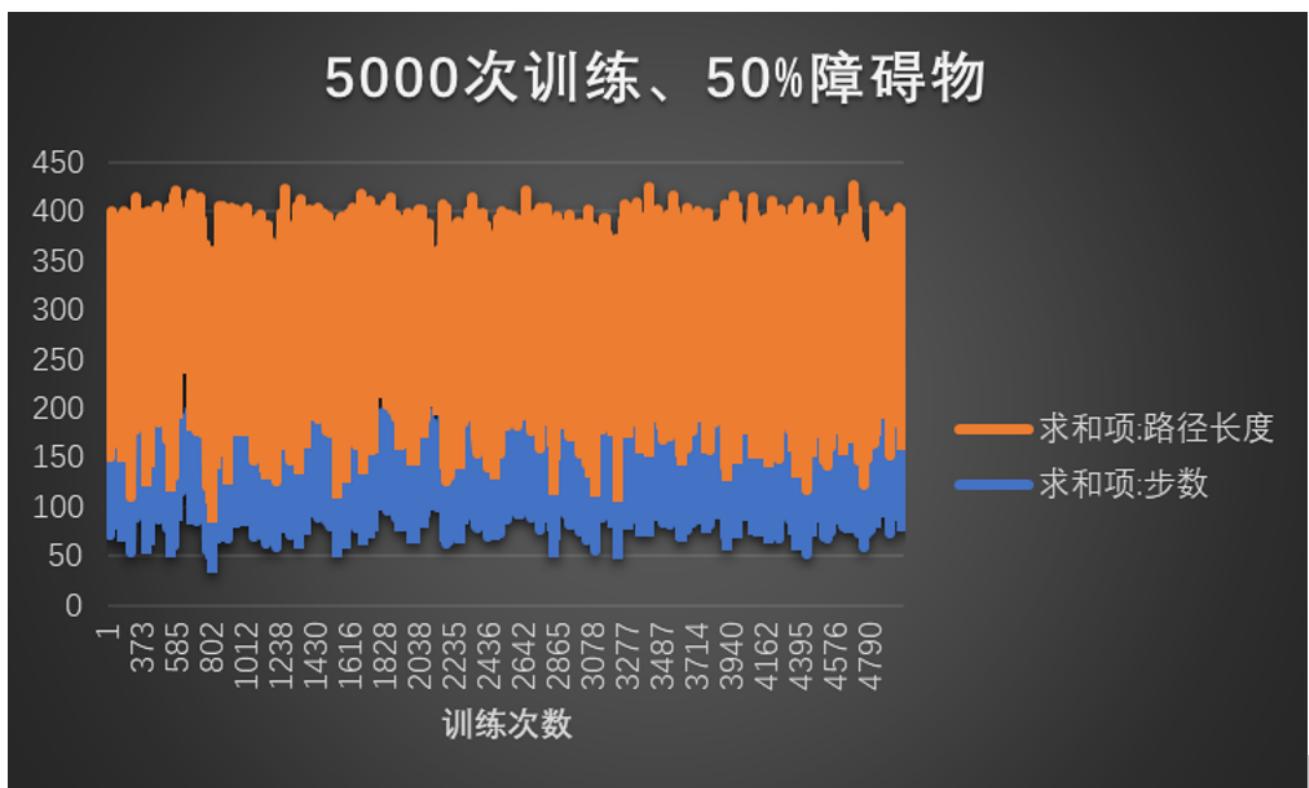


图 4.23 训练 5000 次，障碍物比例为 50% 时的步数和路径长度随成功次数变化的图像

可观察出这样几个现象：

- (1) 相同训练次数的情况下，随着障碍物比例增加，成功率递减。

(2) 随着训练次数的增加，成功的比例也在增大，训练200次时，成功率都在10%以下，障碍物比例为50%时，甚至只有3.5%。训练1000次时，障碍物比例为20%和30%的情况下，成功率都达到50%左右，有大幅度提高，但是障碍物为50%的情况下，却只有1.9%，很意外，训练次数为5000次时，三种障碍物比例情况下，成功率都接近50%，障碍物比例为20%时，成功率达到56%，障碍物比例为50%时，成功率有大幅度提高，达到44%。

(3) 训练次数达到5000次时，算法仍然没有收敛，可能存在问题。

六 . 实践过程、体会与反思

本次是想借着这个课题多学一些东西，包括算法设计技能和代码开发技能，选题结束后，先是用大概一个晚上的时间了解了机器学习的大概知识，在监督学习、无监督学习、强化学习中选择了通过与环境交互来修改自身策略的强化学习算法，阅读了讲解马尔可夫决策过程、Q-learning算法等知识的文献，最后决定选择Q-learning算法来做机器人路径规划，借此来入门人工智能，对创造出的智能体通过探索来自我学习出一套策略的这种思想叹为观止。

然后阅读了一些相关博客，了解他人在开发中如何将Q-learning算法代码化，情景化，应用到具体的路线规划中，因为时间不是很充裕，之前我又没有学习过python与matlab，但我知道这两种语言在人工智能开发中应用最广，且效率很高，但由于现实条件，我只能选择熟悉的C++语言来开发算法的应用框架，通过资料检索，我选择了两篇博客作为主要参考文献，一篇指导开发情景，一篇指导开发算法，应用Qt Creator作为开发工具，开发情景花费了很多时间，大概有四五个整天，因为我基本是从0开始上手QT，中间边开发边遇错，边搜索边学习才把情景开发齐全，后面大概又四五天不断地发现错误并改错，最后在2022年12月16日开始训练算法，开发过程很烧脑，但通过这一个小项目让我对QT开发有了初步了解，增强了代码改错能力，信息搜集能力，也算是有所收获。

训练结果不好，次数少的时候，由于Q-learning算法搜索效率低，成功的次数很少，但是训练次数多了的时候，却又有别的问题，因为我是把每次训练的结果都写入两个txt文件，当我打开文件查看结果时，属实吃了一惊，一些奇奇怪怪的数字让我很困惑是从何而来，但是我又想不出是什么问题，Q表的更新为什么会在若干次训练之后变的非常混乱，是算法的问题，还是电脑的问题，我也不得而知，把数据放到excel生成折线图后会发现成功寻径时的步数和路径长度都是震荡的，而不是呈现出下降后收敛成一条直线的状态（中国矿业大学数学学院的一篇硕士研究生论文（文献[8]）中的仿真结果图像就是这样）。因为寻找错误->搜索资料->修改算法->参数调优->训练结果->撰写报告等等一些工作还会耗费大量的时间，权衡利弊，我决定先停在这里，后续时间充沛时，或者开学遇到老师和同学时再做以后的更改。

七 . 算法改进和研究展望

1. 基于Q-learning算法改进研究现状

很多研究将启发式算法与强化学习算法相结合来改进算法，利用启发式算法的一些优点特性来弥补Q-learning算法的不足，文献[8]—[10]就是相关的例子。

文献[8]针对Q-learning算法前期搜索效率低，规划路线慢甚至找不到目的地的缺点进行改进，作者观察到Q值表与蚁群算法的信息素矩阵结构相同，意义相近，因此作为两种算法的结合点，对基本蚁群算法进行改进后融合入Q-learning算法，得到了很好的效果。文献[9]提出了一种基于萤火虫算法的Q-learning

算法，尝试在加快Q-learning算法收敛速度的同时，避免算法陷入局部最优。首先利用萤火虫算法寻优能力强的特点初始化Q值；其次在Q-learning算法的动作选择过程中，将迭代信息嵌入到选择策略中，设计了混合选择策略，令算法可以动态选择贪婪搜索策略和根据Q值搜索策略，避免算法陷入到局部最优；文献[10]中Liw等针对三维无人机路径规划问题，提出了基于A*和Q-learning算法的混合算法，混合算法中设计了自适应探索因子，以提高改进算法的收敛速度；文献[11]针对Q-learning算法收敛速度慢，学习时间长的问题，提出了一种基于具有先验知识的改进Q-learning移动机器人路径规划算法。该路径规划算法在原来标准Q-learning算法的基础上增加了一层深度学习层、在算法初始化的过程中加入了关于环境的先验知识作为启发信息并且根据不同的状态在构建的标量场中的位置获得不同的奖赏值来使得奖赏值具有导向性，解决了算法学习初始阶段的无目的性，使得算法在学习过程中具有导向性，提高了算法的学习效率以及算法的收敛速度。文献[12]针对移动机器人采用强化学习方法进行路径规划时存在的学习效率低及收敛速度慢等问题，提出一种改进的Q-learning算法。首先提出动态动作集策略，根据机器人当前点与终点的位置来选择其动作集；然后在算法中加入启发式奖惩函数，使得机器人采取不同的动作收获不同的奖励。由此来改进算法，进而提高算法的学习效率，加快算法收敛。文献[13]结合Q-learning算法和Sarsa算法的优点，设计了基于前馈神经网络的Q-learning算法，并将其应用到无人机器人的路径规划问题中；

目前一种将深度学习的感知能力和强化学习的决策能力相结合的算法——深度强化学习，在机器人路径规划中发挥了重要作用，相关研究也层出不穷，文献[14]将前沿的深度强化学习算法应用于移动机器人路径规划问题中，使移动机器人通过强化学习算法在未知环境下进行探索与学习，并训练其决策能力，最终实现连续动作空间下的路径规划与避障。文献[15]将全局路径规划算法和局部路径规划算法进行结合，研究基于深度强化学习的路径规划算法，在此基础上进一步研究多机器人编队控制算法，并在基于ROS平台的轮式机器人上进行实验验证。根据机器人半径对原始地图的障碍物栅格进行扩张，避免规划出的路径紧贴障碍；在评价函数中使用路径长度和转弯角度作为代价值，并使用时间统一计算，以找出一条转折点较少的路径；在找出路径之后通过平滑算法进一步优化，得到更有利于机器人控制的路径。而且研究了基于深度强化学习的局部路径规划算法，提出了改进深度确定性策略梯度（DDPG）算法。

2.本文算法的后期改进展望

文献[16]给我较大启发，该文献剖析了强化学习算法在路径规划中的不足以及改进的一些可行方向。在该开始的学习过程中，智能体（机器人）对环境一无所知，此时如果状态空间很大，奖励值又很稀疏，很多奖励值都初始化为0，那么智能体就会用大量时间作无意义的搜索，因为找到有意义的奖赏值概率很小，这就是强化学习算法收敛非常慢的主要原因。

该文章提出了一个“启发式策略选择”的概念，就是在强化学习中融入人的先验知识或者过程知识来设计和优化启发式奖赏函数，与原有的根据Q值表进行动作选择的策略相融合，来使智能体更快达到最优。

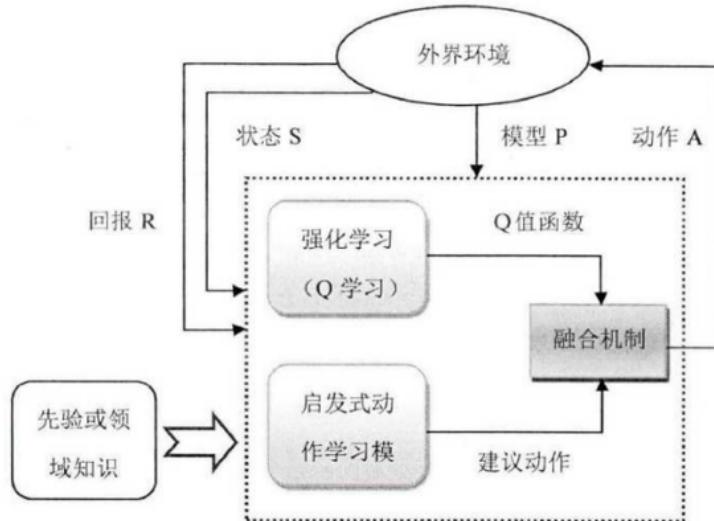


图 7.1 引入启发函数策略的强化学习模型

CSDN @_坐看云起时_

而且做出的动作选择是要有概率性的：

$$\begin{cases} \text{启发函数推荐策略} & rand < \beta \\ \text{根据Q值决定策略} & others \end{cases}$$

CSDN @_坐看云起时_

当生成的随机数小于 β 时，采用我们的启发式策略推荐一个动作，帮助智能体更好的探索，如果随机数大于等于 β ，就让智能体自己根据Q值做出动作，让 β 随着迭代次数逐渐减小，因为Q表逐渐成熟，到后期就不要我们的启发式推荐了。这就相当于大人在儿童年幼时帮助其走路，指引其决策，随着儿童自己的学习，知识体系日趋成熟，大人也就可以减小干预。

我打算后期如果有机会改进时，采用这种策略，启发函数我打算就设置为：做出决策后，会导致下一步格子距离目标点的直线距离最短所对应的那个动作，只是一个构想，后期还需要测试实践。

后续该研究的变化（如果有），我会在我的CSDN博客上更新。

ID : _坐看云起时_ _坐看云起时_的博客_CSDN博客-操作系统,计算机系统与硬件,OJ领域博主

(暂结)

程序、代码存放在：

链接: <https://pan.baidu.com/s/1HhV41zoZWC3V7iAZqVFVAg?pwd=2222>

提取码: 2222

参考文献：

[1]王鹤静,王丽娜.机器人路径规划算法综述[J/OL].桂林理工大学学

报:1-15[2022-12-14].<http://kns.cnki.net/kcms/detail/45.1375.N.20221213.1104.001.html>

[2] 基于可视图法 (VG) 的路径规划算法简述_慕羽★的博客-CSDN博客_vg算法

[3] 向征,赵歆波,曹师好,张宝尚.基于遗传算法的铁路列车图像配准研究[J].中国体视学与图像分析,2021,26(03):253-260.DOI:10.13505/j.1007-1482.2021.26.03.006.

[4] 《人工智能导论》 (第三版) , 丁世飞著

[5] 人工智能学习之机器人路径规划优化_榕林子的博客-CSDN博客_人工智能路径规划

[6] 基于Q-learning的无人机三维路径规划 (含完整C++代码) _~在下小吴的博客-CSDN博客_无人机三维路径规划

[7] qt中connect函数探究_jKingle的博客-CSDN博客

[8] 柯逸. 基于蚁群Q-learning算法的移动机器人路径规划[D].中国矿业大学,2022.DOI:10.27623/d.cnki.gzkyu.2022.000545.

[9] 王付宇,张康,谢昊轩,陈梦凯.基于改进Q-learning算法的移动机器人路径优化[J].系统工程,2022,40(04):100-109.

[10] D. Li, W. Yin, W. E. Wong, M. Jian and M. Chau, "Quality-Oriented Hybrid Path Planning Based on A* and Q-Learning for Unmanned Aerial Vehicle," in IEEE Access, vol. 10, pp. 7664-7674, 2022, doi: 10.1109/ACCESS.2021.3139534.

[11] 王慧,秦广义,夏鹏,杨春梅,王刚.基于改进强化学习算法的移动机器人路径规划研究[J].计算机应用与软件,2022,39(07):269-274.

[12] 潘国倩,周新志.基于启发式强化学习的移动机器人路径规划算法研究[J].现代计算机,2022,28(10):57-61.

[13] Wang Y H , Li T , Lin C J . Backward Q-learning: The combination of Sarsa algorithm and Q-learning[J]. Engineering Applications of Artificial Intelligence, 2013, 26(9):2184-2193.

[14] 吴亚琦. 基于深度强化学习的移动机器人自主路径规划研究[D].华东交通大学,2021.DOI:10.27147/d.cnki.ghdju.2021.000229.

[15] 周盛世. 基于深度强化学习的机器人路径规划算法研究[D].南京理工大学,2021.DOI:10.27241/d.cnki.gnjgu.2021.002288.

[16] 马朋委. Q_learning强化学习算法的改进及应用研究[D].安徽理工大学,2016.