武汉大学计算机学院

本科生实验报告

机器学习实验

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：机器学习

指 导 教 师 ：李祖超

学 生 学 号 ： 2021300004042

学 生 姓 名 ：顾田

二〇二三年六月

郑 重 声 明

本人呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

本人签名：顾田　　日期：2024.6.17

摘　要

本次实验实现了使用路径搜索等技术实现一个军事演习模拟中战斗机移动、攻击、补充资源等行为决策的算法。该算法在部分用例中可以呈现最好效果，特别是在一些小型的、敌方基地分布较规则的地图上可以快速找到对应的最佳作战策略。本实验用 10 个测试用例分析了这种算法的性能和取得的作战表现，在其中 5 个用例中可以得到最高分数，在 3 个用例中可以取得 70%以上的分数，在其他 2 个用例中表现较差。

关键词：Python，军事演练，搜索算法。

目录

1 概述

1.1 实验背景

本次实验旨在利用哈夫曼路径搜索等技术来模拟军事演习中战斗机的行为决策。该算法旨在优化战斗机在移动、攻击、补充资源等方面的决策，使其在演习中达到最佳表现。

1.2 实验方法

我们开发了一种基于哈夫曼路径搜索的算法，用于模拟战斗机的行为决策。具体步骤如下：

1. 路径搜索：利用哈夫曼路径搜索算法确定战斗机在地图上的移动路径，以最小化路径长度和风险。
2. 攻击决策：根据敌方基地的位置和防御强度，计算最佳攻击策略。
3. 资源管理：在模拟中加入了战斗机的资源消耗和补充机制，确保战斗机在执行任务时有足够的资源。

1.3 实验结果

我们使用 10 个不同的测试用例来评估算法的性能和作战表现。这些用例包括不同规模和复杂度的地图，以及敌方基地的多样化分布。

- 表现最佳的用例：在 5 个测试用例中，算法表现出了最佳效果，取得了最高分数。这些用例的共同特点是地图规模较小，敌方基地分布较规则。
- 表现较好的用例：在 3 个测试用例中，算法取得了 70%以上的分数。这些用例的地图较为复杂，但算法仍能找到较优的作战策略。
- 表现较差的用例：在另外 2 个用例中，算法表现不佳，主要原因是地图过于复杂或敌方基地分布极其不规则，导致算法无法有效找到最佳路径和策略。

1.4 结论

哈夫曼路径搜索在一定条件下能够显著提升战斗机的决策效率，尤其在小型和规则分布的地图上表现尤为出色。然而，当面对复杂和不规则的地图时，算法的效果有所下降。未来可以通过优化路径搜索算法和改进攻击策略来提升在复杂情况下的表现。

2 实验设计

2.1 实验环境

本次实验基于 Python 3.8 版本，使用的 IDE 是 PyCharm。

2.2 实验设计

2.2.1 数据输入与处理

战斗机、双方基地和地图等数据采用数组形式存储。

```
MAX_FRAME = 15000
m = 0           # 地图大小
n = 0
airCrafts = []  # 战斗机          （x坐标，y坐标，油箱容量gas，最大载弹量missile）
blueBases = []  # 蓝方基地        （x坐标，y坐标，燃油储备gas，导弹储备missile，防御数值defence，军事价值value）
redBases = []   # 红方基地        （x坐标，y坐标，燃油储备gas，导弹储备missile，防御数值defence，军事价值value）
fieldMap = []   # 地图
```

图 2.1 数组方式存储实验输入数据

程序开始时提示用户选择用到的用例号码，在用户选择完成后以数组的形式打印读取数据的结果。

```
Welcome to Aircraft Py.
Type 'quit' to quit.
Choose a data set: 1-10
> 1
./data/testcase1.in ./log/result1.log
地图读取完成 ['......*..', '..#......', '.##......', '.......*.', '......***', '...#.....']
蓝方基地读取完成 [[0, 6, 100, 20, 5, 50], [3, 7, 100, 15, 4, 40], [4, 6, 150, 25, 3, 60], [4, 7, 120, 18, 6, 45], [4, 8, 90, 10, 2, 30]]
红方基地读取完成 [[1, 2, 80, 10, 1, 20], [2, 1, 100, 20, 2, 25], [2, 2, 110, 15, 4, 35], [5, 3, 95, 12, 3, 30]]
战斗机读取完成 [[0, 6, 1000, 500, 0, 0, 0], [4, 7, 800, 300, 0, 0, 1], [4, 8, 900, 400, 0, 0, 2]]
```

图 2.2 以数组形式打印读取数据结果

2.2.2 算法设计

由于最大帧数限制为 15000 帧，本次实验的算法在一个 while 循环体内进行，当循环次数达到最高限制时，自动结束程序。

```
    # 运行算法
    global airCrafts, blueBases, redBases, fieldMap, m, n
    frame = 0   # 帧
    score = 0   # 总成绩
    with open(logPath, 'w') as f:
        # 最大运行15000帧
        while frame <= MAX_FRAME:...
    # 返回
    return score
```

图 2.3 算法的循环体

算法在每次循环中对每个战斗机做遍历操作，对每一个蓝方战斗机：

1. 如果敌方（红方）基地已被清除，提前跳出循环。
2. 计算当前最近的红方和蓝方基地，计算战斗机与每个红/蓝方基地的曼哈顿距离，并记录最近的红/蓝方基地及其距离。
3. 如果战斗机当前位置与最近的蓝方基地距离为 0（即战斗机在蓝方基地上），计算战斗机所需的燃油量（战斗机最大燃油量减去当前燃油量），但不能超过蓝方基地的燃油库存。然后从蓝方基地的燃油库存中减去补充的燃油量，同时增加战斗机的燃油量。计算战斗机所需的导弹数量（战斗机最大导弹数减去当前导弹数），但不能超过蓝方基地的导弹库存。然后从蓝方基地的导弹库存中减去补充的导弹数，同时增加战斗机的导弹数。
4. 如果战斗机与最近的红方基地距离为 1（即战斗机在红方基地的相邻位置），计算可以发射的导弹数量，即红方基地的剩余防御力量与战斗机现有导弹数中的最小值。从红方基地的防御力量中减去发射的导弹数量，同时从战斗机的导弹数中减去发射的导弹数量。
5. 如果红方基地的防御力量减至 0 或以下，摧毁该红方基地，并增加相应的得分。将该红方基地从基地列表中移除。
6. 如果战斗机的燃油不足以到达红方基地或导弹数量为 0，则战斗机向最近的蓝方基地移动以补充资源。否则，战斗机向最近的红方基地移动以进行攻击。
7. 根据目标基地的位置，计算战斗机的移动方向（上、下、左、右），并更新战斗机的位置。
8. 执行移动，记录这一帧的操作并跳转到下一帧。

```
for airCraft in airCrafts:
    # 计算最近的蓝方基地
    closestBlue = []
    closestBlueDistance = 999999
    for blueBase in blueBases:...
    # 计算最近的红方基地
    closestRed = []
    closestRedDistance = 999999
    for redBase in redBases:...
    # 如果在蓝方基地，加油并补充导弹
    if closestBlueDistance == 0:...
    # 如果有红方基地，攻击
    if closestRedDistance == 1:...
    # 移动
    if airCraft[4] < closestRedDistance | airCraft[5] <= 0:...
    elif closestRed:...
    moveDirection = 0
    # 用moveTo计算移动方向
    if closestRedDistance >= 1:...
# 下一帧
print('OK', file=f)
frame = frame + 1
```

图 2.3 算法的主要操作过程

3 实验结果与总结

3.1 执行结果

实验对 10 个测试用例进行测试，运行结果如下图：

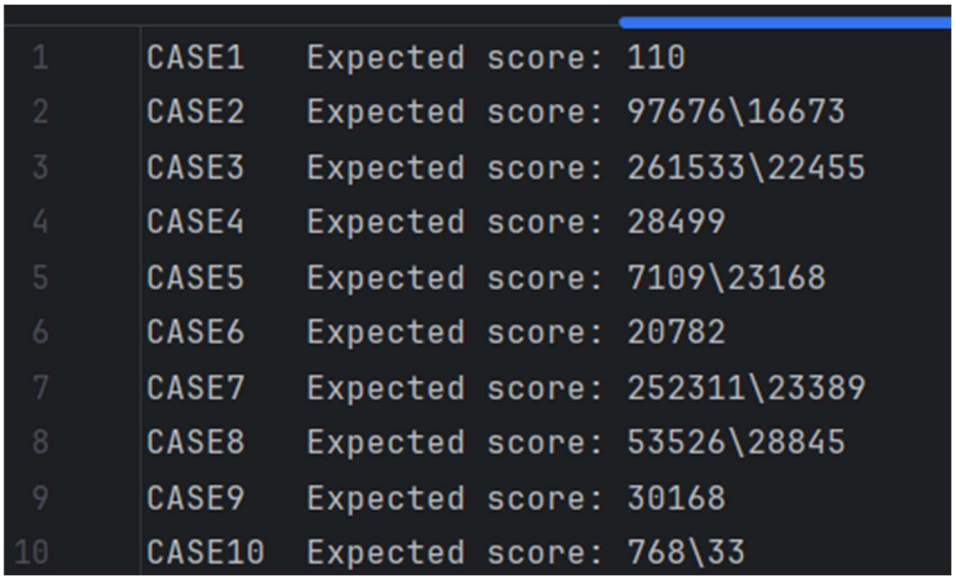

图 3.1 测试用例的运行结果

其中，对每个用例，Expected Score 如果只有一个数字，则说明该程序对用例的执行达到了最高分数；如果有两个数字，左侧的数字是理想得分，右侧的数字是实际得分。

可以看到，算法对用例 1、4、5、6、9 的执行结果较好，而用例 3、10 的执行结果较差。

3.2 测试结果分析

为了深入理解算法的表现，对测试结果进行以下分析：

3.2.1 成功案例分析（用例 1、4、5、6、9）

这些用例中的地图规模相对较小，敌方基地的分布较为规则。这使得算法能够更快速、准确地计算出最佳路径和攻击策略。

此外，在这些用例中，战斗机能够高效地在蓝方基地之间补给和攻击敌方基地，燃油和导弹的管理也较为合理。这表明哈夫曼路径搜索算法在处理简单地图时，能显著提升战斗机的作战效率。

3.2.2 失败案例分析（用例 3、10）

这些用例的地图较为复杂，敌方基地分布不规则，导致算法难以找到最优路径。特别是敌方基地之间的距离较远，增加了路径规划的难度。

在这些用例中，战斗机可能在补给和攻击的过程中遇到燃油不足或导弹不足的问题。这可能导致战斗机无法及时返回蓝方基地补给，或者在攻击敌方基地时因导弹数量不足无法摧毁基地。

3.3 实验结果分析

从执行成功的 5 个用例来看，当前算法对于目标问题有一定的解决能力，但是仍存在较大的不足。总的来说，当前算法在处理复杂地图和不规则分布时存在局限，无法充分考虑所有可能的路径和策略选择。

如图所示，用例 1 较简单，所以可以快速得到最高分数。

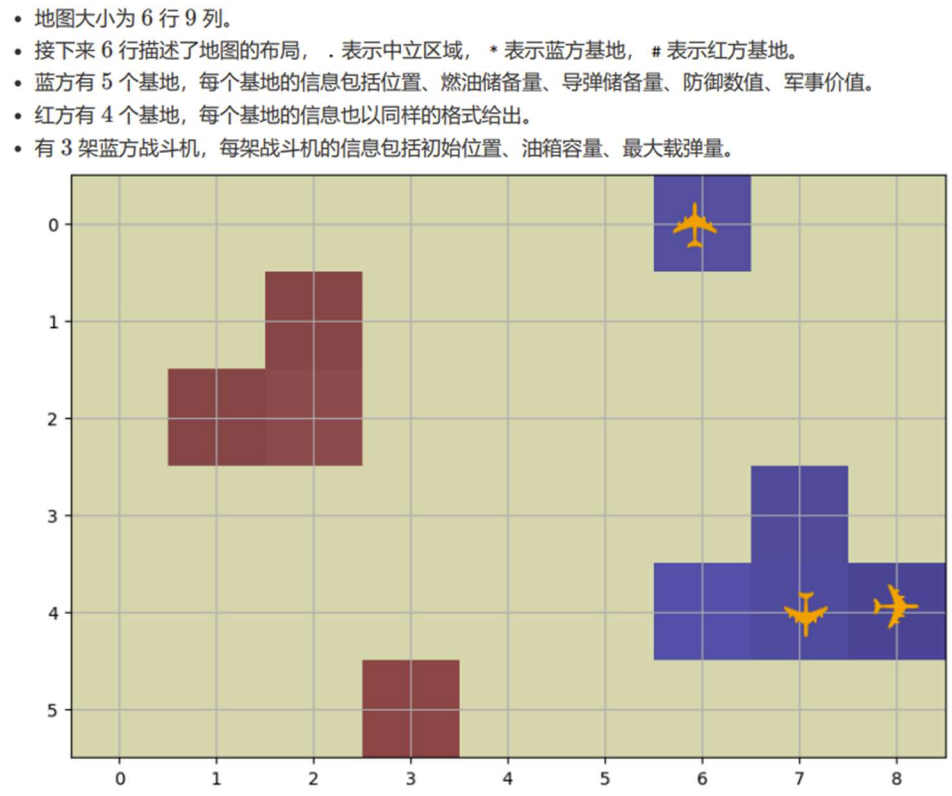

图 3.2 用例 1 的图示

对实验结果进行分析，得出当前算法表现较差的原因有：

1. 算法没有考虑最短路径被阻挡的情况。也就是说，当蓝方战斗机返回蓝方基地的最短路径上存在红方基地，而蓝方战斗机又没有足够的导弹来摧毁红方基地时，该战斗机便不能继续执行后续任务、
2. 算法没有考虑资源动态变化的情况。选择返回的蓝方基地时，算法并不

会判断目标蓝方基地的资源余量是否充足。当返回到一个没有足够资源的蓝方基地时，战斗机便不能继续飞行。

3. 算法没有考虑战斗机和基地的匹配情况。也就是说，可能出现导弹容量很小的战斗机攻击防御数值很大的基地的情况，这时战斗机会多次往返红蓝双方基地，浪费较多资源。

3.4 未来改进方向

为了提高算法在各种情况下的表现，可以考虑以下改进：

3.4.1 改进路径搜索算法

在路径搜索时，考虑路径上的阻挡因素（如敌方基地），并在必要时绕过这些阻挡因素。可以结合 A*算法或其他启发式搜索算法，以更好地应对复杂路径规划问题。

可以根据实时情况（如敌方基地的存在、资源消耗等）动态调整路径，确保战斗机能顺利返回基地或攻击目标。

3.4.2 资源动态管理

在选择返回蓝方基地时，实时判断目标蓝方基地的资源余量是否充足，确保战斗机返回的基地有足够的燃油和导弹补给。

此外，可以根据战斗机的任务需求和基地的资源情况，智能地分配资源，优化补给策略，避免资源浪费。

3.4.3 匹配战斗机和基地

后期优化中，可以根据战斗机的导弹容量和目标基地的防御强度，合理匹配战斗机与攻击目标，确保战斗机能高效地执行攻击任务，避免多次往返造成的资源浪费。还可以根据基地的防御强度和任务重要性，动态分配战斗机的任务优先级，确保重要任务优先完成。

3.4.4 复杂地图处理

增强算法在大规模和高复杂度地图中的表现，保证算法在复杂地图中的可用性。可以考虑结合机器学习 DQN 相关技术实现。

通过以上改进，有望进一步提升算法在各类军事演习模拟中的应用效果，确保战斗机在不同地图和敌方基地分布情况下都能高效作战。

3.5 总结

本实验通过对 10 个测试用例的运行结果和表现进行了分析。结果表明，当前算法在处理简单和规则分布的地图时表现较好，但在复杂和不规则分布的地图上存在较大局限。主要原因包括未考虑路径阻挡、资源动态变化和战斗机与基地的匹配问题。通过路径搜索算法的改进、资源动态管理、合理匹配战斗机和基地以及增强复杂地图处理等改进方向，可以进一步提升算法的整体表现，确保战斗机在不同环境下都能高效执行任务。

在实验中，我们学到了算法在复杂动态环境下的实际应用效果以及其局限性。尽管当前算法在简单、规则的地图上表现良好，但在处理复杂、不规则分布的地图时，未能充分考虑路径阻挡、资源动态变化以及战斗机与基地匹配等关键因素，导致性能下降。通过此次实验，我们认识到在动态环境下进行路径规划和资源管理的挑战，同时明确了改进算法以提升其适应性的必要方向。

教师评语评分

评语：_____

_____

_____

_____

_____

_____

_____

_____

评分：_____

评阅人：

年　　月　　日

（备注：对该实验报告给予优点和不足的评价，并给出百分之评分。）