

**课程实验报告**

|  |  |
| --- | --- |
| 课程名称 | 自动规划 |
| 实验名称 | 赛道问题的PDDL求解 |

|  |  |
| --- | --- |
| 学 号： | 58122231 |
| 姓 名： | 陆文韬 |
| 学 院： | 人工智能学院 |
| 专 业： | 人工智能 |
| 指导教师： | 张志政 |
| 实验日期： | 2024-2025秋季学期 |

1. 规划问题定义
2. 流的描述：

本程序实现了一个基于PDDL（Planning Domain Definition Language）的赛车轨道规划问题。问题的核心是控制一辆赛车在8×8的网格轨道上从起始位置移动到目标区域，同时考虑赛车的速度状态和障碍物约束。

问题采用状态空间搜索的方法，其中每个状态包含赛车的位置坐标和当前速度向量。状态转换通过两类基本动作实现：速度调整和位置移动。这种建模方式很好地捕捉了赛车运动的物理特性，即速度的连续性和惯性。

1. 动作描述：

系统定义了两个核心动作：

**速度调整动作（adjust-velocity）**：允许赛车在当前位置调整其速度向量。前置条件要求赛车必须在指定位置且具有当前速度状态，同时新速度必须是允许的调整范围内（每个方向上可以增减1或保持不变）。效果是更新赛车的速度状态，移除旧速度并设置新速度。

**移动动作（move-car）**：根据当前速度移动赛车位置。前置条件包括赛车必须在起始位置、具有指定速度、存在合法移动路径且目标位置不是墙壁。效果是更新赛车位置，从起始位置移动到目标位置。

这两个动作的设计体现了赛车运动的真实约束：必须先调整速度再移动，且移动距离和方向由速度决定。

1. 初始环境：

初始状态设定赛车位于网格位置p74（对应坐标(7,4)），初始速度为(1,1)，即向右上方移动。

环境包含完整的8×8网格，位置编码为pXY格式，其中X和Y分别表示行列坐标。速度范围设定为-2到+2的整数值，提供了丰富的运动可能性。

关键的约束条件包括：允许的速度调整规则（每次调整幅度不超过1）、各种速度下的合法移动路径、以及障碍物设置。特别地，在网格中间设置了一道水平障碍墙（位置p42到p45），增加了规划的复杂性。

1. 目标环境：

目标是让赛车到达指定的目标区域，具体包括三个位置：p03、p04、p05（对应坐标(0,3)、(0,4)、(0,5)）。

目标条件采用了存在量词表达，即只要赛车到达任意一个目标位置即可，不要求特定的最终速度状态。这种设计提供了灵活性，允许多种解决方案。

从空间布局看，目标区域位于网格的上方，而起始位置在下方，需要赛车进行较长距离的导航，同时避开中间的障碍物。

1. 规划生成机制（规划算法）介绍：

本程序基于经典的STRIPS规划框架，采用前向状态空间搜索方法。算法从初始状态开始，通过应用可行动作生成后继状态，直到找到满足目标条件的状态。

搜索过程中，算法维护一个状态空间图，其中节点表示系统状态，边表示动作转换。每个状态由赛车位置和速度向量唯一确定。算法需要处理两类动作的交替应用：先调整速度，再执行移动。

为了提高效率，算法可能采用启发式搜索策略，如A\*算法，使用曼哈顿距离或欧几里得距离作为启发函数估计到目标的代价。

1. 规划程序设计与实现
2. 介绍
   1. 编程语言：

本程序使用PDDL（Planning Domain Definition Language）标准语言编写。PDDL是人工智能规划领域的标准建模语言，采用类似Lisp的S-表达式语法。程序分为两个主要部分：领域定义（domain）和问题实例（problem）。

* 1. 运行环境：

程序需要在支持PDDL的规划求解器上运行，常见的包括FF（Fast-Forward）、LAMA、Fast Downward等。这些求解器通常运行在Unix/Linux环境下，支持命令行调用。

1. 运行情况说明
   1. 输入输出说明：

**输入**：程序接受两个文件作为输入，领域文件定义了问题的通用结构（谓词、动作模式），问题文件定义了具体的实例（对象、初始状态、目标状态）。

**输出**：求解器输出一个动作序列，表示从初始状态到目标状态的完整解决方案。输出格式通常包括每个动作的名称和参数，以及执行代价。

* 1. 主要数据结构说明：

**状态表示**：每个状态由一组事实（facts）表示，包括位置事实car-at和速度事实car-vel。状态空间的大小为位置数×速度组合数。

**动作表示**：动作通过参数化的动作模式表示，包含前置条件和效果。系统在运行时将动作模式实例化为具体的动作实例。

**搜索图**：算法维护一个搜索图结构，节点存储状态信息，边存储动作转换。图的构建采用惰性求值策略，只在需要时扩展节点。

* 1. 运行案例：

一个典型的解决方案可能包含以下步骤：

1. 从位置p74开始，当前速度(1,1)
2. 调整速度或直接移动到p63
3. 继续按对角线路径移动：p52→p41→p30
4. 最终到达目标位置p03

整个过程需要大约7-10个动作步骤，具体取决于速度调整的策略。

1. 分析
2. 通用性说明（从算法和规划语言两方面说明）：

**算法通用性**：程序采用标准的STRIPS规划框架，具有很好的通用性。算法可以处理任意规模的网格、不同的速度范围、各种障碍物配置。通过修改问题文件，可以轻松适应不同的赛车轨道布局。

**规划语言通用性**：PDDL语言的标准化保证了程序的可移植性。程序可以在不同的PDDL求解器上运行，无需修改代码。同时，PDDL的表达能力足够处理复杂的约束和目标条件。

1. 完备性说明：

算法具有完备性，即如果存在解决方案，算法一定能找到。这是因为：

1. 状态空间是有限的（64个位置×25种速度组合）
2. 算法采用系统性搜索，不会遗漏任何可能的状态
3. 动作的定义保证了状态转换的确定性

但需要注意，完备性的前提是问题确实存在解，且搜索空间在合理范围内。

1. 计算复杂度说明：

**时间复杂度**：最坏情况下为O(b^d)，其中b是分支因子（平均每个状态的后继数量），d是解的深度。在本问题中，b约为10-15（考虑速度调整和移动的组合），d约为10-20步。

**空间复杂度**：主要由搜索图的存储需求决定，为O(|S|)，其中|S|是访问过的状态数量。在最坏情况下可能达到O(64×25)=O(1600)。

实际复杂度会因启发式搜索和剪枝技术而显著降低。

1. 正确性说明：

程序的正确性通过以下机制保证：

1. **语义正确性**：动作的前置条件和效果正确地模拟了赛车运动的物理约束
2. **约束满足**：所有移动都经过可行性检查，确保不会穿越障碍物
3. **目标验证**：解决方案必须满足明确定义的目标条件

程序设计中的关键正确性保证包括速度调整的限制、移动路径的合法性检查、以及目标条件的精确定义。

1. 最优性：

当前程序不保证解的最优性，因为：

1. 使用的是满足性规划，而非优化规划
2. 没有定义明确的代价函数
3. 搜索策略可能找到第一个可行解就停止

要获得最优解，需要引入代价函数（如步数、时间、能量消耗）并使用最优化搜索算法。

1. 不足说明：

程序存在以下主要不足：

**效率问题**：对于更大规模的问题，搜索空间会指数级增长，可能导致性能问题。缺乏高效的启发式函数和剪枝策略。

**建模局限**：当前模型较为简化，没有考虑加速度限制、燃料约束、时间窗口等更复杂的现实约束。

**扩展性限制**：硬编码的网格大小和速度范围限制了程序的通用性。增加新的约束类型需要重新设计领域模型。

**求解器依赖**：程序的性能很大程度上依赖于所使用的PDDL求解器的效率和优化策略。这些不足为未来的改进工作指明了方向，包括引入更高效的算法、更精确的建模、以及更好的用户接口设计。