中山大学计算机学院

人工智能

本科生实验报告

（2023 学年春季学期）

课程名称：Artificial Intelligence

教学班级	**刘咏梅老师班级**	专业（方向）	**信息与计算科学**
学号	**22336313**	姓名	**郑鸿鑫**

一、 实验题目

1. 补全 blocks.pddl 完成积木世界规划问题

2. 用规划完成 15puzzle 中四个问题之一

二、 实验内容

1. 算法原理

PDDL（Planning Domain Definition Language）

- 定义：PDDL 是一种用于形式化描述规划问题的标准化语言。
- 组成：包括域（domain）和问题（problem）两个部分，其中域定义了规划的语言结构，问题定义了具体的规划实例。

FF 规划器（Fast Forward Planner）

- 方法：采用启发式搜索算法，通过前向搜索在状态空间中寻找解决方案。
- 特点：不保证得到最优解，但通常能高效找到可行解。

搜索策略

- 状态空间：利用紧凑的状态表示法（如 STAR）高效存储和操作状态。
- 动作选择：基于当前状态和动作效果选择合法动作。
- 目标测试：当状态满足所有目标条件时，停止搜索。

解决方案提取

- 路径回溯：从搜索过程中提取出满足目标状态的动作序列。

2. 伪代码

Block.pddl 中两个动作的伪代码：

```
    // 伪代码：blocks 域的 move 动作
procedure move(x, y)
    requires
        // 动作前提条件
        x 是一个物体且在桌子上 (ontable x)
        y 是一个物体且在桌子上 (ontable y)
        x 不是 y (确保 x 和 y 是不同的物体)
        x 上面无积木 (clear x) 表示 x 上没有其他物体
        y 上面无积木 (clear y) 表示 y 上没有其他物体
    modifies
        // 动作效果
        x 现在在 y 上 (on x y)
        x 上面有积木 (not (clear x))
        y 变得清晰 (clear y)
        对于所有物体 z
            如果 x 原来在 z 上 (on x z)
                则 x 不再在 z 上 (not (on x z))
                且 z 变得清晰 (clear z)
    end requires
    end modifies
    do
        // 执行移动操作
        将 x 从其当前位置移动到 y 上
        更新状态以反映 x 现在在 y 上
        更新状态以反映 x 上面有积木
        更新状态以反映 y 现在上面无积木
        对于所有物体 z
            如果 x 原来在 z 上
                则更新状态以反映 x 不再在 z 上
                且更新状态以反映 z 现在上面无积木
    end do
end procedure
// 伪代码：blocks 域的 moveToTable 动作
procedure moveToTable(x)
    requires
        // 动作前提条件
        x 是一个物体
        x 不在桌子上 (not (ontable x))
        x 上面无积木 (clear x) 表示 x 上没有其他物体
    modifies
        // 动作效果
        x 现在在桌子上 (ontable x)
        x 上面有积木 (not (clear x))
        对于所有物体 z
            如果 x 原来在 z 上 (on x z)
                则 x 不再在 z 上 (not (on x z))
                且 z 变得清晰 (clear z)
    end requires
    end modifies
```

```
    do
        // 执行移动到桌子的操作
        将 x 移动到桌子表面
        更新状态以反映 x 现在在桌子上
        更新状态以反映 x 上面有积木
        对于所有物体 z
            如果 x 原来在 z 上
                则更新状态以反映 x 不再在 z 上
                且更新状态以反映 z 现在上面无积木
    end do
end procedure
```

Puzzle.pddl 伪代码：

```
// 伪代码：PPT1 问题的初始状态和目标状态
// 定义位置类型和数字类型
type loc
type num
// 定义位置和数字对象
object L1, L2, ..., L16 of type loc
object B1, B2, ..., B15, empty of type num
// 初始状态 (:init)
// 每个数字块的初始位置
is_at(B1, L7)
is_at(B2, L9)
is_at(B3, L10)
...
is_at(empty, L4) // 表示空格的初始位置
// 相邻位置的初始状态
adjacent(L1, L2)
adjacent(L2, L3)
...
adjacent(L16, L15)
// 目标状态 (:goal)
// 目标是将每个数字块放置到对应的编号位置，并且空格位于最后的位置
goal(and(
  is_at(B1, L1),
  is_at(B2, L2),
  ...
  is_at(B15, L15),
  is_at(empty, L16),
  is_empty(L16) // 确保空格的位置是空的
))
```

Puzzle.pddl 伪代码：

```
// 伪代码：puzzle 域的 slide 动作
// 定义两种类型：数字（num）和位置（loc）
type num
type loc
// 定义谓词
// is-at ?n - num ?l - loc：表示数字块 ?n 位于位置 ?l
// adjacent ?l1 ?l2 - loc：表示位置 ?l1 与位置 ?l2 相邻
```

```
// is-empty ?l - loc：表示位置 ?l 是空的
procedure slide(n, l1, l2)
    requires
        // 动作前提条件
        is-at(n, l1) // 数字块 n 必须在位置 l1 上
        is-empty(l2)  // 位置 l2 必须为空
        adjacent(l1, l2) // 位置 l1 和 l2 必须相邻
        not(= l1 l2) // 位置 l1 和 l2 不能相同

    modifies
        // 动作效果
        is-at(n, l2) // 数字块 n 现在在位置 l2 上
        is-empty(l1) // 位置 l1 现在为空
        not (is-empty l2)  l2 不再是空的

    do
        // 执行滑动操作
        move_block(n, l1, l2) // 实际移动数字块 n 从 l1 到 l2
        update_state(l1, true) // 更新状态以反映 l1 现在为空
        update_state(l2, false) // 更新状态以反映 l2 不再为空
end procedure
```

3. 关键代码展示（带注释）

Block.pddl 源代码

```
(define (domain blocks)
  (:requirements :strips :typing:equality
                 :universal-preconditions
                 :conditional-effects
                 :negative-preconditions)
  (:types physob);定义参数类型
  (:predicates  ;定义谓词
       (ontable ?x - physob) ;x 在桌子上
       (clear ?x - physob)  ;x 上无其他积木
       (on ?x ?y - physob)) ;x 在 y 上面
  (:action move
            :parameters (?x ?y - physob)  ;定义动作中的参
数          :precondition (and(clear ?x)(clear ?y)(not(= ?x ?y)))
            ;前提：x 上无积木，y 上无积木，x 和 y 是两个不同的积木
            :effect (and(on ?x ?y)(clear ?x)(not(clear ?y))
                    (forall (?z - physob)
                      (when (on ?x ?z)
                        (and(not(on ?x ?z))(clear ?z))
                      )
                    )
                    )
;效果：x 在 y 上，x 上无积木，y 上有积木，且如果原来 x 下有积木 z，则 z 上现在无积
木
            )
  (:action moveToTable
             :parameters (?x - physob) ;定义动作中的参数
```

```
            :precondition (and (not(ontable ?x)) (clear ?x))
            ;前提：x 不在桌子上，x 上无积木
            :effect (and(ontable ?x)(clear ?x)
                   (forall (?z - physob)
                     (when (on ?x ?z)
                       (and(not(on ?x ?z))(clear ?z))
                     )
                   )
                   )
  ;效果：x 在桌子上，x 上无积木，且如果原来 x 下有积木 z，则 z 上现在无积木
  )
)
```

Puzzle.pddl 源代码

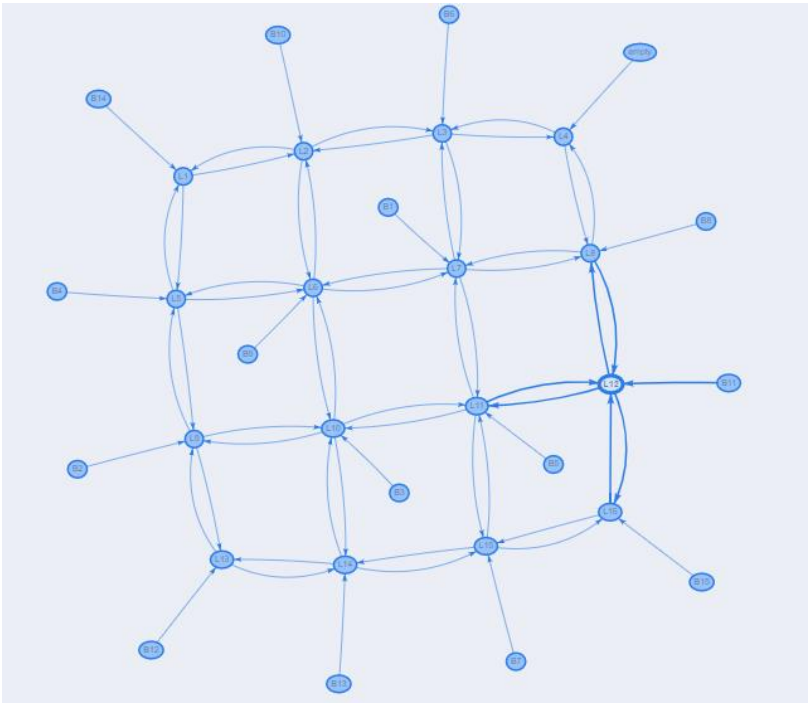
```
(define (domain puzzle)
  (:requirements :strips :equality :typing)
  (:types num loc) ;定义类型，数字块，位置
  (:predicates ;定义谓词
    (is-at ?n - num ?l - loc);n 在位置 l 处
    (adjacent ?l1 ?l2 - loc);l1 和 l2 位置相邻
    (is-empty ?l - loc));位置 l 为空
  (:action slide
    :parameters (?n - num ?l1 ?l2 - loc);定义动作中的参数
    :precondition (and (is-at ?n ?l1)(is-
empty ?l2)(adjacent ?l1 ?l2)(not(= ?l1 ?l2)))
    ;前提：n 在 l1 处，l2 是空的，l1 和 l2 相邻，l1 和 l2 不相等
    :effect (and (is-at ?n ?l2)(is-empty ?l1)(not (is-empty ?l2))))
    ;效果：n 在 l2 位置，l1 位置为空，l2 不再为空
)
```

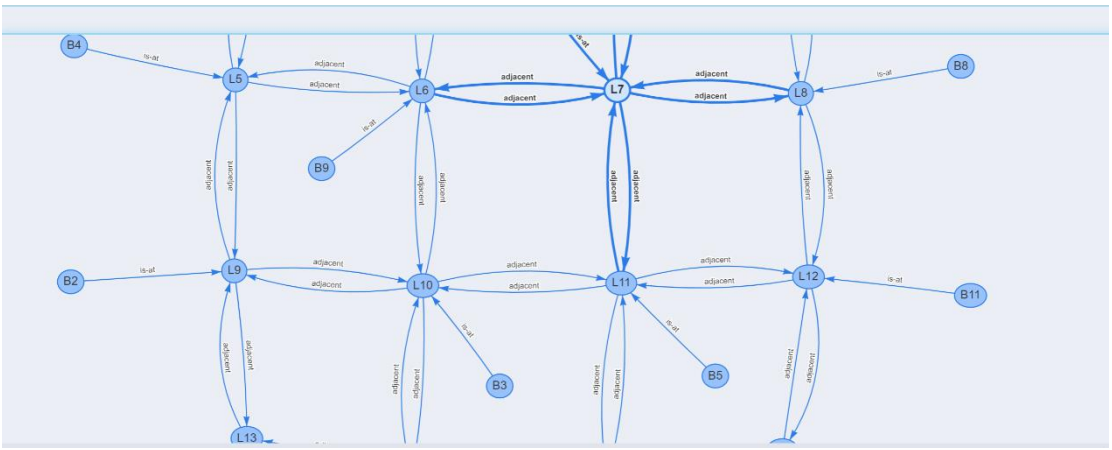
PPT1.pddl

```
(define (problem PPT1)
  (:domain puzzle)
  (:objects L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 - loc
            B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15 empty - num)
  (:init (is-at B1 L7)(is-at B2 L9)(is-at B3 L10)(is-at B4 L5)
        (is-at B5 L11)(is-at B6 L3)(is-at B7 L15)(is-at B8 L8)
        (is-at B9 L6) (is-at B10 L2)(is-at B11 L12)(is-at B12 L13)
        (is-at B13 L14)(is-at B14 L1)(is-at B15 L16)(is-at empty L4)
        (is-empty L4)
        ;定义初始状态
         (adjacent L1 L2)(adjacent L2 L3)(adjacent L3 L4)(adjacent L4 L8)
         (adjacent L1 L5)(adjacent L2 L6)(adjacent L3 L7)(adjacent L5 L6)
         (adjacent L5 L9)(adjacent L6 L7)(adjacent L6 L10)(adjacent L7 L8)
         (adjacent L7 L11)(adjacent L8 L12)(adjacent L9 L10)(adjacent L13 L9)
         (adjacent L10 L11)(adjacent L10 L14)(adjacent L11 L12)(adjacent L11 L15)
         (adjacent L12 L16)(adjacent L13 L14)(adjacent L14 L15)(adjacent L16 L15)
        ;由于谓词是单向的，但是相邻状态时相互的，所以需要交换顺序后重新给出
         (adjacent L2 L1)(adjacent L3 L2)(adjacent L4 L3)(adjacent L8 L4)
         (adjacent L5 L1)(adjacent L6 L2)(adjacent L7 L3)(adjacent L6 L5)
         (adjacent L9 L5)(adjacent L7 L6)(adjacent L10 L6)(adjacent L8 L7)
         (adjacent L11 L7)(adjacent L12 L8)(adjacent L10 L9)(adjacent L9 L13)
         (adjacent L11 L10)(adjacent L14 L10)(adjacent L12 L11)(adjacent L15 L11)
         (adjacent L16 L12)(adjacent L14 L13)(adjacent L15 L14)(adjacent L15 L16))
  (:goal (and (is-at B1 L1)(is-at B2 L2)(is-at B3 L3)(is-at B4 L4)(is-at B5 L5)
              (is-at B6 L6)(is-at B7 L7)(is-at B8 L8)(is-at B9 L9) (is-at B10 L10)
              (is-at B11 L11)(is-at B12 L12)(is-at B13 L13)(is-at B14 L14)(is-at B15 L15)
              (is-at empty L16)(is-empty L16)))
        ;定义目标状态
```

```
)
```

注意到其中对于 **adjacent** 谓词，表示的是两个位置之间的相互关系，所以需要重复给出，以得到网络如下所示：

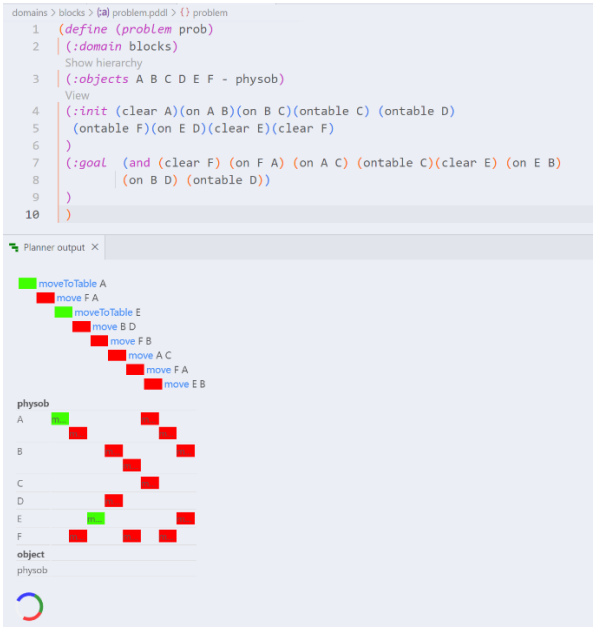




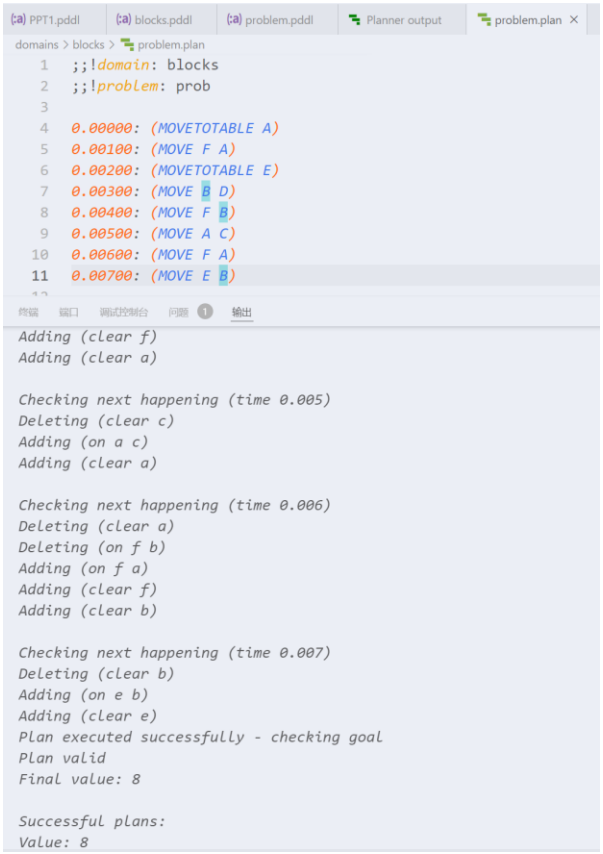
三、 实验结果及分析

1. 实验结果展示示例

实验任务 **1**：积木世界



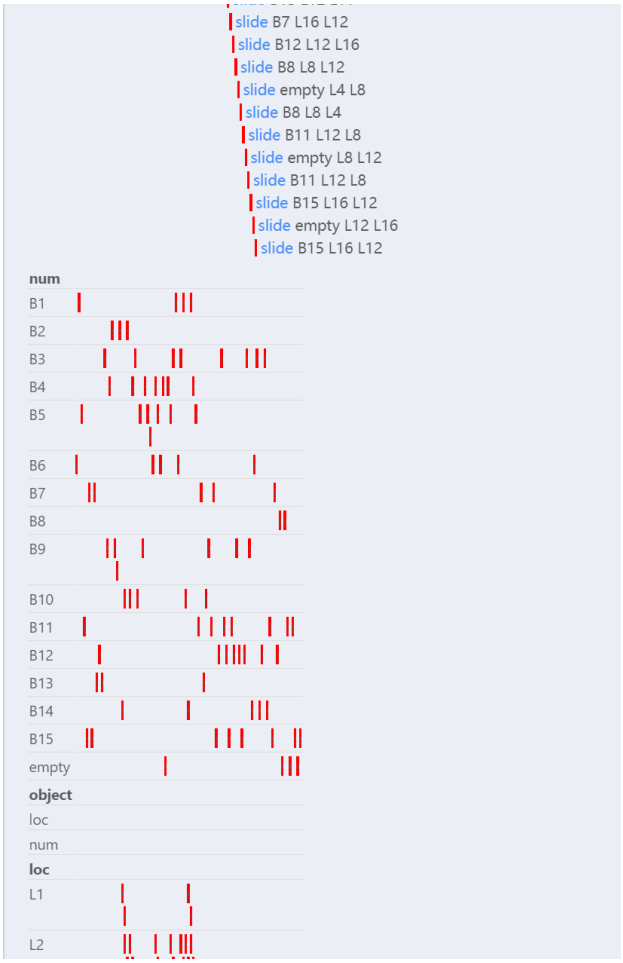
经过检验，该 Plan 可以将初始状态转化为目标状态：



实验任务 2：15puzzle 中的 PPT1 样例：

由于实验结果较长，故放在 Result 文件夹中，此处只给出部分截

图：



经过检验，该 Plan 可以将初始状态成功转化为最终状态：

domains > puzzle > PPT1.plan

```
;;!domain: puzzle
;;!problem: PPT1

0.00000: (SLIDE B6 L3 L4)
0.00100: (SLIDE B1 L7 L3)
0.00200: (SLIDE B5 L11 L7)
0.00300: (SLIDE B11 L12 L11)
0.00400: (SLIDE B15 L16 L12)
0.00500: (SLIDE B7 L15 L16)
0.00600: (SLIDE B15 L16 L15)
0.00700: (SLIDE B7 L15 L16)
0.00800: (SLIDE B13 L14 L15)
0.00900: (SLIDE B12 L13 L14)
0.01000: (SLIDE B13 L14 L13)
0.01100: (SLIDE B3 L10 L14)
```

终端 端口 调试控制台 问题 1 输出

```
Checking next happening (time 0.086)
Deleting (is-empty l12)
Adding (is-at b15 l12)
Adding (is-empty l16)

Checking next happening (time 0.087)
Deleting (is-empty l16)
Adding (is-at empty l16)
Adding (is-empty l12)

Checking next happening (time 0.088)
Deleting (is-empty l12)
Adding (is-at b15 l12)
Adding (is-empty l16)
Plan executed successfully - checking goal
Plan valid
Final value: 89

Successful plans:
Value: 89
```

四、 参考资料

1. 解析 HTTP 错误码 400 Bad Request 及其常见原因与解决方法_http status 400 – bad request-CSDN 博客
2. PDDL Domain - Planning.wiki - The AI Planning & PDDL Wiki