Part III-B: Artificial Intelligence Outline

Lecture by 熊庆宇
Note by THF

2024 年 10 月 17 日

目录

Lecture 7

10.14

0.1 状态空间表示法

Notation. 回忆：

命题 → 谓词 → 产生式 → 框架 → 状态

Definition. 状态空间表示法：表示问题及其搜索过程

Example. 与空格相连的棋子可以移动到空格中：

表 1: 初始状态

2	8	3
1		4
7	6	5

如何将某一初始状态变成目标状态：

表 2: 目标状态

1	2	3
8		4
7	6	5

Example. 渡河问题：三个传教士 M 和三个野人 C 过河，只有一条能装下两个人的船，在河的一方或船上，如果野人的人数大于传教士的人数，那么传教士会有危险，如何使所有人安全地过河

$$\text{状态空间适用的场景}\begin{cases}\text{调度}\\ \text{分配}\\ \text{导航}\\ \text{路径规划}\\ \text{游戏}\\ \cdots\end{cases}.$$

Notation. 状态空间法主要包括：

1. 状态集：其中的每个元素表示一种状态
2. 操作算符集：连结状态间的条件
3. 状态空间：包括状态集、操作算符集、目标状态集

Example. 某棋局：

表 3: 棋盘 S

X_1	X_2	X_3
X_8	X_0	X_4
X_7	X_6	X_5

用 $S = (X_0, X_1, \ldots, X_8)$ 表示状态，0 代表空格

如：表 6 表示为：$S_0 = (0, 2, 8, 3, 4, 5, 6, 7, 1)$

表 7 表示为 $S_8 = (0, 1, 2, 3, 4, 5, 6, 7, 8)$

Lecture 7

表 4: S1

2	8	3
1	4	
7	6	5

将表 6 中棋子 4 左移，状态变为：$S_1 = (4, 2, 8, 3, 0, 5, 6, 7, 1)$

继续移动，直至找到一套操作得到状态 S_8：$S_0 \to S_1 \to \ldots \to S_8$

该问题称为八数码难题

Notation. 八数码难题的算符：

仅为空格制定操作：空格上下左右移动，空格的约束条件为不能移出棋盘

表 5: S0->S1

2		3
1	8	4
7	6	5

$\xrightarrow{\text{空格向左移}}$

	2	3
1	8	4
7	6	5

Example. 此时空格有 3 种移动方式

Notation. 状态空间图：

把初始状态可达到的各状态所组成的空间设想为由各种状态对应的节点组成的图（有向图）

图的节点表示状态

图的边表示操作算符

Example. 二阶汉诺塔问题：

$S_i\,(a, b)$ 表示状态盘 A 在 a 柱上，盘 B 在 b 柱上

算符：$A\,(i, j)$ 表示将 A 盘从 i 柱移动到 j 柱

$B\,(i, j)$ 同理

汉诺塔问题的状态图可以是双向图，限制为：A 不能在 B 下方

Example. 渡河问题：

初始状态：$(0, 0, 0)$

Lecture 7

目标状态：$(3,3,1)$

状态格式：(右岸传教士数量, 右岸野人数量, 船的位置)

算符：

Move-1m1c-lr：将一个传教士和一个野人从左边传到右边

Move-2c-lr：将两个野人从左边移到右边

Move-1m-rl：将一个传教士从右边移动到左边

操作：

$(0,0,0) \xrightarrow{\text{Move-1m1c-lr}} (1,1,1) \xrightarrow{\text{Move-1c-rl}} (1,0,0) \xrightarrow{\text{Move-2c-lr}} (1,2,1) \xrightarrow{\text{Move-1c-rl}} (1,1,0) \to \dots$

作业：第 26-29 题

1 搜索求解策略

Notation. 早期搜索策略：图搜索、盲目搜索、启发式搜索

高级搜索技术：规则演绎系统、产生式系统

Definition. 搜索技术：根据问题的实际情况，不断寻找可利用的知识，构造出一条代价较少的推理路线

搜索技术是 AI 的基本技术之一

搜索好的标准：

1. 搜索空间小
2. 解最佳

Example. 爬山路径：

1. 问题全状态空间：整座山
2. 搜索空间：山的路
3. 解：爬山路径

$$\text{一般搜索算法}\begin{cases}\text{无信息搜索（盲目搜索）}\begin{cases}\text{宽度/广度优先搜索（BFS）}\\ \text{深度优先搜索（DFS）}\\ \text{等代价搜索}\end{cases}\\ \text{有信息搜索}\begin{cases}\text{A 算法}\\ \text{A* 算法}\end{cases}\end{cases}.$$

Notation. 搜索算法：

必须记住哪些节点已经遍历（OPEN 表）

需给出下一步可以选择哪些节点（CLOSED 表）

必须记住从目标节点返回的路径

Notation. BFS: Breath-First Search

首先扩展根节点

然后扩展根节点的**所有后继节点**

以此类推，在第 n 层节点未完全遍历之前不进入第 $n+1$ 层的遍历

Lecture 8

10.17

Notation. DFS: Deep-First Search

优先朝长节点扩展：先进入开放表的节点先扫描

优点：搜索空间可以远小于宽度优先

缺点：忽略深度

修正：加入深度界限，在已知目标节点的深度范围时限制搜索深度

最坏情况：$o(n)$

应用：状态表 = 树状图

Notation. 等代价搜索/Dijkstra 算法：

BFS 的一种推广

$g(n)$ 代表从初始节点到节点 n 的代价

$c(n_1, n_2)$ 表示从 n_1 到 n_2 的代价

$g(n_2) = g(n_1) + c(n_1, n_2)$

优点：加入了状态图中的路径长短元素（走一步看一步），以等代价选择下一节点的选择