计算机科学188：人工智能导论
2024年春季 笔记3

作者（其他所有笔记）：尼基尔·夏尔马
作者（贝叶斯网络笔记）：乔希·胡格和杰基·梁，由王瑞吉编辑
作者（逻辑笔记）：亨利·朱，由考佩林编辑
致谢（机器学习与逻辑笔记）：部分章节改编自教材《人工智能：一种现代方法》。

最后更新时间：2023年8月26日

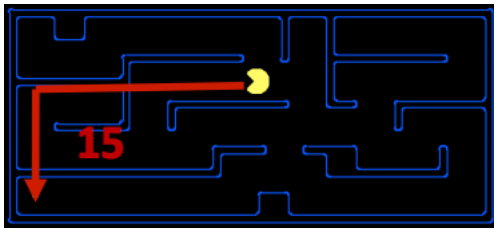
启发式搜索

一致代价搜索很好，因为它既完备又最优，但它可能相当慢，因为在从起始状态搜索目标时它会向各个方向扩展。如果我们对应该集中搜索的方向有一些概念，我们可以显著提高性能并更快地"锁定"目标。这正是启发式搜索的重点。

启发式方法

启发式方法是用于估计到目标状态距离的驱动力——它们是将一个状态作为输入并输出相应估计值的函数。这种函数所执行的计算特定于正在解决的搜索问题。出于我们将在下面的A*搜索中看到的原因，我们通常希望启发式函数是到目标的剩余距离的下限，因此启发式方法通常是松弛问题（其中原始问题的一些约束已被去除）的解决方案。回到我们的吃豆人示例，让我们考虑前面描述的路径规划问题。用于解决此问题的一种常见启发式方法是曼哈顿距离，对于两点 (x_1, y_1) 和 (x_2, y_2)，其定义如下：

$$Manhattan(x_1, y_1, x_2, y_2) = |x_1 - x_2| + |y_1 - y_2|$$



上述可视化展示了曼哈顿距离有助于解决的松弛问题——假设吃豆人想要到达迷宫的左下角，在假设迷宫中没有墙壁的情况下，它计算从吃豆人的当前位置到其期望位置的距离。这个距离是松弛搜索问题中的精确目标距离，相应地也是实际搜索问题中的估计目标距离。借助启发式方法，在我们的智能体中实现逻辑变得非常容易，这使得它们在决定执行哪个动作时能够"倾向于"扩展那些估计更接近目标状态的状态。

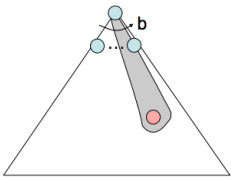
这种偏好概念非常强大，以下两种实现启发式函数的搜索算法会用到它：贪婪搜索和A*算法。

贪婪搜索

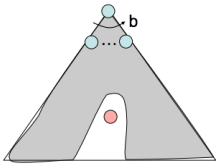
•描述 - 贪婪搜索是一种探索策略，总是选择启发式值最低的前沿节点进行扩展，这对应于它认为最接近目标的状态。

•前沿表示 - 贪婪搜索的操作与一致代价搜索相同，采用优先队列前沿表示。不同之处在于，贪婪搜索不是使用计算出的反向代价（到达该状态路径上边权的总和）来分配优先级，而是使用启发式值形式的估计正向代价。

•完备性和最优性 - 如果存在目标状态，贪婪搜索不一定能找到它，也不是最优的，特别是在选择了非常糟糕的启发式函数的情况下。它在不同场景下的行为通常相当不可预测，可能直接找到目标状态，也可能像引导不佳的深度优先搜索一样探索所有错误的区域。



(a) 某一天的贪婪搜索 :)



(b) 糟糕日子里的贪心搜索 :(

A* 搜索

•描述 - A* 搜索是一种探索策略，它总是选择估计总成本最低的前沿节点进行扩展，其中总成本是从起始节点到目标节点的全部成本。

•前沿表示 - 与贪心搜索和一致代价搜索一样，A* 搜索也使用优先队列来表示其前沿。唯一的区别在于优先级选择方法。A* 通过将这两个值相加，将一致代价搜索中使用的总反向成本（到该状态路径上边权重的总和）与贪心搜索中使用的估计正向成本（启发值）相结合，有效地得出从起始点到目标点的估计总成本。鉴于我们希望最小化从起始点到目标点的总成本，这是一个绝佳的选择。

•完备性和最优性 - 给定一个合适的启发函数（我们稍后会讲到），A* 搜索既是完备的又是最优的。它结合了我们迄今所讨论的所有其他搜索策略的优点，兼具贪心搜索通常的高速度以及一致代价搜索的最优性和完备性！

可采纳性和一致性

既然我们已经讨论了启发式算法以及它们在贪心搜索和A*搜索中的应用方式，那我们花些时间来讨论一下什么才是一个好的启发式算法。为此，我们首先用以下定义重新阐述在一致代价搜索、贪心搜索和 A^* 中用于确定优先级队列排序的方法：

- $g(n)$ - 由一致代价搜索计算出的表示总反向代价的函数。
- $h(n)$ - 贪心搜索使用的启发式值函数，即估计的正向代价。
- $f(n)$ - 由 A^* 搜索使用的表示估计总成本的函数。 $f(n) = g(n) + h(n)$ 。

在探讨什么构成"好的"启发式算法这个问题之前，我们必须首先回答一个问题：无论我们使用何种启发式函数，A*是否都能保持其完备性和最优性。实际上，很容易找到会破坏这两个令人向往的属性的启发式算法。例如，考虑启发式函数 $h(n) = 1 - g(n)$ 。无论搜索问题是什么，使用这种启发式算法都会产生

$$\begin{aligned} f(n) &= g(n) + h(n) \\ &= g(n) + (1 - g(n)) \\ &= 1 \end{aligned}$$

因此，这样的启发式算法会将A*搜索简化为广度优先搜索（BFS），其中所有边的代价都是相等的。正如我们已经表明的，在边权重不恒定的一般情况下，广度优先搜索并不能保证是最优的。

使用 A^* 树搜索时最优性所需的条件称为可采纳性。可采纳性约束指出，可采纳启发式算法估计的值既不是负数也不是高估。将 $h^*(n)$ 定义为从给定节点 n 到达目标状态的真正最优向前代价，我们可以将可采纳性约束用数学公式表示如下：

$$\forall n,\ 0 \leq h(n) \leq h^*(n)$$

定理。对于给定的搜索问题，如果启发式函数 h 满足可采纳性约束，那么在该搜索问题上使用带有 h 的A* 树搜索将产生最优解。

证明。假设对于给定的搜索问题，在搜索树中存在两个可达的目标状态，一个最优目标 A 和一个次优目标 B 。由于从初始状态可以到达 A ，所以 A 的某个祖先节点 n （可能包括 A 本身）当前必定在前沿。我们使用以下三个陈述来证明 B 之前会选择 n 进行扩展：

1. $g(A) < g(B)$ 。因为 A 被给定为最优，而 B 被给定为次优，所以我们可以得出结论， A 到初始状态的反向成本低于 B 。

2. $h(A) = h(B) = 0$ ，因为我们已知我们的启发式方法满足可采纳性约束。由于 A 和 B 都是目标状态，从 A 或 B 到目标状态的真正最优成本就是 $h^*(n) = 0$ ；因此 $0 \leq h(n) \leq 0$ 。

3. $f(n) \leq f(A)$ ，因为通过 h, $f(n) = g(n) + h(n) \leq g(n) + h^*(n) = g(A) = f(A)$ 的可采纳性。经过节点 n 的总成本至多是 A 的真实反向成本，而这也是 A 的总成本。

我们可以结合陈述1.和2.得出如下结论： $f(A) < f(B)$

$$f(A) = g(A) + h(A) = g(A) < g(B) = g(B) + h(B) = f(B)$$

将上述推导的不等式与陈述3.相结合的一个简单结果如下：

$$f(n) \leq f(A) \wedge f(A) < f(B) \Longrightarrow f(n) < f(B)$$

因此，我们可以得出结论， n 在 B 之前展开。由于我们已经针对任意的 n 证明了这一点，所以我们可以得出结论， A 的所有祖先（包括 A 本身）都在 B 之前展开。

我们上面发现的树搜索的一个问题是，在某些情况下，它可能永远找不到解决方案，在状态空间图中陷入无限搜索同一个循环。即使在我们的搜索技术不涉及这种无限循环的情况下，通常也会多次重新访问同一个节点，因为有多种方式可以到达同一个节点。这会导致工作量呈指数级增加，自然的解决方案是简单地记录哪些状态已经展开，并且不再展开它们。更明确地说，在使用你选择的搜索方法时，维护一个已展开节点的“已到达”集合。然后，在展开之前确保每个节点不在该集合中，如果不在，则在展开后将其添加到该集合中。带有此附加优化的树搜索称为图搜索 [1] ，其伪代码如下所示：

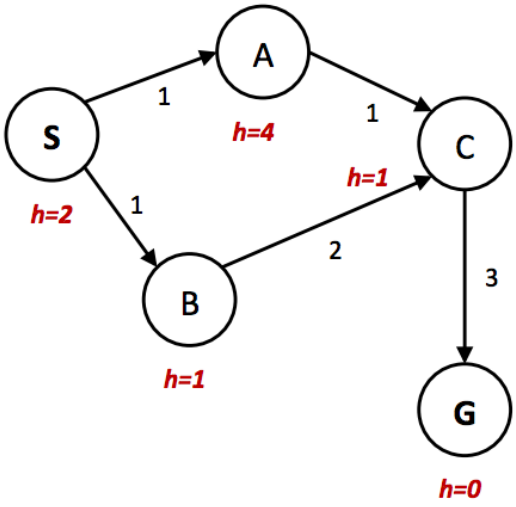
[1] 在其他课程中，比如CS70和CS170，你可能在图论的背景下接触过“树”和“图”。具体来说，树是一种满足特定约束条件（连通且无环）的图。但这并不是我们在本课程中所讨论的树搜索和图搜索之间的区别。

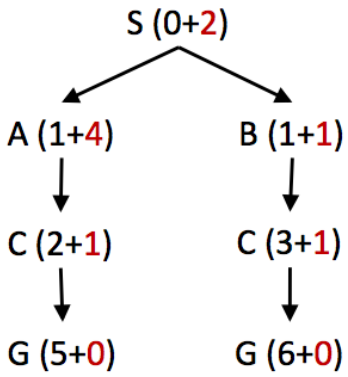
函数GRAPH-SEARCH（问题，前沿）返回一个解决方案或失败

reached ← an empty set
frontier← INSERT(MAKE-NODE(INITIAL-STATE[problem]), frontier) **while** ***not*** *IS-EMPTY(frontier)* **do**
end
node ← POP(frontier) **if** *problem.IS-GOAL(node.STATE)* **then**
end
return node
if *node.STATE is not in reached* **then**
end
add node.STATE in reached
for *each child-node in EXPAND(problem, node)* **do**
end
frontier ← INSERT(child-node, frontier)

return failure

请注意，在实现中，将已到达的集合存储为不相交集合而不是列表至关重要。将其存储为列表需要花费 $O(n)$ 操作来检查成员资格，这消除了图搜索旨在提供的性能提升。图搜索的另一个注意事项是，即使在采用可接受启发式的情况下，它也往往会破坏 A* 的最优性。考虑以下简单的状态空间图和相应的搜索树，标注了权重和启发式值：





在上述示例中，很明显最优路径是沿着 $S \to A \to C \to G$，总路径成本为 $1+1+3=5$。到达目标的唯一其他路径 $S \to B \to C \to G$ 的路径成本为 $1+2+3=6$。然而，由于节点 A 的启发式值比节点 B 的启发式值大得多，节点 C 首先作为节点 B 的子节点沿着第二条次优路径展开。然后它被放入"已到达"集合中，因此当 A* 图搜索将其作为 A 的子节点访问时，未能重新展开它，所以它从未找到最优解。因此，为了在 A* 图搜索下保持最优性，我们需要一个比可采纳性更强的属性，

一致性。一致性的核心思想是，我们不仅要确保启发式函数低估从任何给定节点到目标的总距离，还要确保图中每条边的成本/权重。启发式函数所衡量的边的成本仅仅是两个相连节点的启发式值之差。从数学角度来看，一致性约束可以表述如下：

$$\forall A, C \quad h(A) - h(C) \leq cost(A, C)$$

定理。对于给定的搜索问题，如果启发式函数 h 满足一致性约束，那么在该搜索问题上使用带有 h 的A* 图搜索将得到最优解。

证明。为了证明上述定理，我们首先证明当使用一致的启发式函数运行 A^* 图搜索时，每当我们移除一个节点进行扩展时，我们就已经找到了到该节点的最优路径。

利用一致性约束，我们可以证明沿着任何路径的节点的 $f\left(n\right)$ 值是不减的。定义两个节点 n 和 n'，其中 n' 是 n 的子节点。那么：

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + cost(n, n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

如果对于沿着一条路径的每一对父子关系 (n, n')，$f\left(n'\right) \geq f\left(n\right)$，那么必然的情况是，沿着该路径 $f\left(n\right)$ 的值是非递减的。我们可以检查上述图在 $f\left(A\right)$ 和 $f\left(C\right)$ 之间违反了这条规则。有了这个信息，我们现在可以证明，每当一个节点 n 被移除用于扩展时，其最优路径就已经被找到。假设与之矛盾的情况是这是错误的——即当 n 从前沿中被移除时，找到的到 n 的路径是次优的。这意味着在前沿上必然存在 n, n'' 的某个祖先，它从未被扩展但却在到 n 的最优路径上。矛盾！我们已经表明沿着一条路径 f 的值是非递减的，所以 n'' 会在 n 之前就被移除用于扩展。

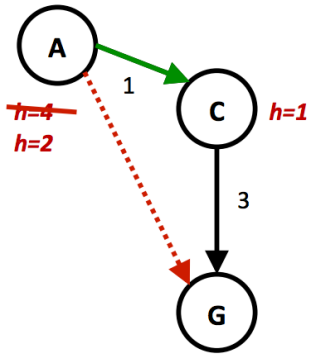
为完成证明，我们剩下要展示的就是，在任何次优目标 B 之前，最优目标 A 总会被取出用于扩展并返回。这是显而易见的，因为 $h\left(A\right) = h\left(B\right) = 0$，所以

$$f(A) = g(A) < g(B) = f(B)$$

就如同我们在可采纳性约束下对A*树搜索最优性的证明一样。因此，我们可以得出结论，在一致启发式下A*图搜索是最优的。I

在继续讨论之前，上述讨论中有几个重要要点：对于可采纳/一致的启发式要有效，根据定义，对于任何目标状态 G，必须满足 $h\left(G\right) = 0$。此外，一致性不仅仅是比可采纳性更强的约束，一致性意味着可采纳性。这仅仅源于这样一个事实，即如果没有边成本被高估（如一致性所保证），从任何节点到目标的总估计成本也不会被高估。

以如下三节点网络为例，说明一个可采纳但不一致的启发式：



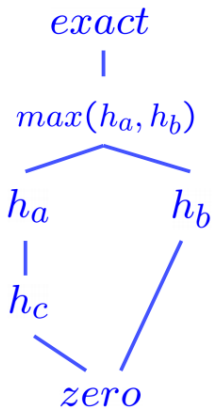
红色虚线对应于估计的总目标距离。如果 $h(A) = 4$，那么该启发式算法是可采纳的，因为从 A 到目标的距离是 $4 \geq h(A)$，对于 $h(C) = 1 \leq 3$ 也是如此。然而，从 A 到 C 的启发式成本是 $h(A) - h(C) = 4 - 1 = 3$。我们的启发式算法估计 A 和 C 之间边的成本为 3，而真实值是 $\text{cost}(A, C) = 1$，一个更小的值。由于 $h(A) - h(C) \nleq \text{cost}(A, C)$，此启发式算法不一致。然而，对 $h(A) = 2$ 运行相同的计算会产生 $h(A) - h(C) = 2 - 1 = 1 \leq \text{cost}(A, C)$。因此，使用 $h(A) = 2$ 会使我们的启发式算法一致。

优势

既然我们已经确定了可采纳性和一致性的属性及其在保持 A^* 搜索最优性方面的作用，我们就可以回到最初创建"好的"启发式算法的问题，以及如何判断一种启发式算法是否比另一种更好。对此的标准度量是优势。如果启发式算法 a 优于启发式算法 b，那么对于状态空间图中的每个节点， a 的估计目标距离大于 b 的估计目标距离。用数学表示为，

$$\forall n : h_a(n) \geq h_b(n)$$

支配性非常直观地捕捉了一种启发式方法优于另一种的概念——如果一个可接受/一致的启发式方法支配另一个，那么它一定更好，因为它总是能更精确地估计从任何给定状态到目标的距离。此外，平凡启发式方法被定义为 $h(n) = 0$，使用它会将A*搜索简化为一致代价搜索。所有可接受的启发式方法都支配平凡启发式方法。平凡启发式方法通常被纳入搜索问题的半格基础中，它位于该半格支配层次结构的底部。下面是一个包含各种启发式方法的半格示例 h_a, h_b，以及 h_c，范围从底部的平凡启发式方法到顶部的精确目标距离：



一般来说，应用于多个可采纳启发式函数的最大值函数也总是可采纳的。这仅仅是因为启发式函数针对任何给定状态输出的所有值都受可采纳性条件 $0 \leq h(n) \leq h^*(n)$ 的约束。此范围内数字的最大值必然也落在同一范围内。对于多个一致的启发式函数，同样也能很容易地证明这一点。对于任何给定的搜索问题，通常会生成多个可采纳/一致的启发式函数，并计算它们输出值的最大值，以生成一个比所有这些函数都更具优势（因此更好）的启发式函数。

搜索：总结

在本笔记中，我们讨论了搜索问题及其组成部分：状态空间、一组动作、转移函数、动作成本、起始状态和目标状态。智能体通过其传感器和执行器与环境进行交互。智能体函数描述了智能体在所有情况下的行为。智能体的合理性意味着智能体试图最大化其期望效用。最后，我们使用PEAS描述来定义我们的任务环境。关于搜索问题，可以使用多种搜索技术来解决，包括但不限于我们在CS 188中学习的五种：

- 广度优先搜索
- 深度优先搜索
- 一致代价搜索
- 贪婪搜索
- A* 搜索

上面列出的前三种搜索技术是无信息搜索的示例，而后两种是有信息搜索的示例，它们使用启发式方法来估计目标距离并优化性能。
我们还对上述技术的树搜索和图搜索算法进行了区分。