COMP9414: 人工智能讲座 2a:问题的解决

韦恩-沃布克
电邮：w.wobcke@unsw.edu.au

本讲座

- 搜索是一种具有广泛适用性的解决问题的 "弱方法"
- 不知情的搜索方法（不使用特定问题的信息）。
- **知情的搜索方法（使用启**发式方法来提高效率）。

激励性的例子

- 你在罗马尼亚度假，在阿拉德，需要去布加勒斯特
- 要解决这个问题，你还需要什么信息？
- 一旦你掌握了这些信息，你该如何解决问题？
- **你怎么知道你的解决方案有多好？**为了评估你的解决方案的质量，你需要哪些额外的信息？

状态空间搜索问题

- 状态空间--
  从初始状态到任何行动序列可达到的所有状态的集合。
- 初始状态 - 状态空间的元素
- 过渡期

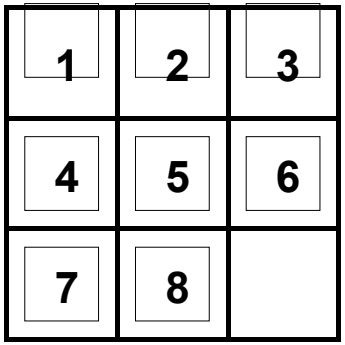
- 操作者 -
  代理人可支配的可能行动的集合；描述在当前状态下执行行动后达到的状态，或
- 继承函数 -

s(*x*)=通过执行一个动作可从状态*x*到达的状态集

- 目标状态 - 状态空间的一个或多个元素
- 路径成本 -
  用于评估解决方案的一连串转换的成本（适用于优化问题）。

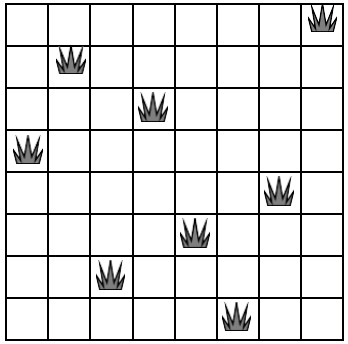
新南威尔士大学
©W.Wobcke等人，2019-2022年

例题--8字谜



状态：八块瓷砖的位置加上空白的位置
操作者：将空白向左、向右、向上、向下移动
目标状态：瓷砖依次排列的状态
路径成本：每一步的成本是1

例题--N-Queens



状态。0到N个皇后排列在棋盘上

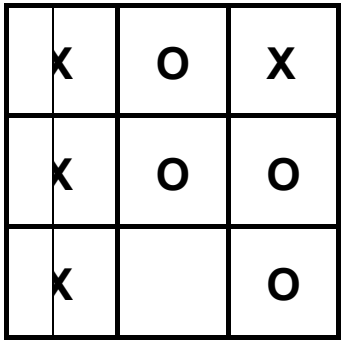
现实世界的问题

- 寻找路线--**机器人**导航、航空旅行计划、计算机/电话网络
- **旅行推**销员问题--规划自动电路板钻头的移动
- VLSI布局 - 设计硅芯片
- **装配**顺序 – **安排复**杂物体的装配，制造过程控制
-

这些都是优化问题，但数学（运筹学）技术并不总是有效。

问题表述--井字形

操作者：把皇后放在空位上
目标状态。棋盘上有N个皇后，没有被攻击 路径成本：0



状态：Os和X在3X3网格上的排列

操作者：将X（O）放在空方格中

目标状态：连续三个X（Os）

路径成本：0

新南威尔士大学
©W.Wobcke等人，2019-2022年

井字游戏--第一次尝试

1	2	3
4	5	6
7	8	9

董事会。0=空白；1=X；2=O
主意。使用有3个^9 =19683个元素的移动表
算法。认为棋盘是一个三元数；转换为十进制；访问移动表；更新棋盘
- 快速；大量内存；费力；不可扩展

井字游戏--第二次尝试

1	2	3
4	5	6
7	8	9

董事会。2=空白；3=X；5=O
算法。每一步都有单独的策略。
目标测试（如果行在下一步给出了胜利）：计算数值的乘积X：

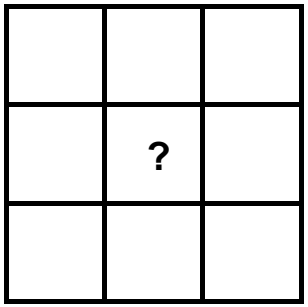
井字游戏--第三次尝试

8	3	4
1	5	9
6	7	2

棋盘是一个神奇的方块!
算法。如同尝试2，但要检查是否获胜--跟踪玩家的"方块"。如果15的差值和两个方格的和≤0或>9，那么这两个方格就不是相邻的。否则，如果与差值相等的方块是空的，则移动到那里。
- 这说明人类解决问题的方式与计算机相比有什么不同？

测试乘积=18（3×3×2）；O：测试乘积=50（5×5×2）。
- **没有1的速度快；内存少得多；更容易理解和掌握；策略事**先确定；不具有可扩展性

井字游戏--第四次尝试



棋盘：由下一步棋产生的棋盘位置列表；对导致胜利的位置的可能性的估计

算法：看每一步棋所产生的位置；选择 "最佳 "的棋。

- 速度较慢；可以处理大量的各种问题

新南威尔士大学
©W.Wobcke等人，2019-2022年

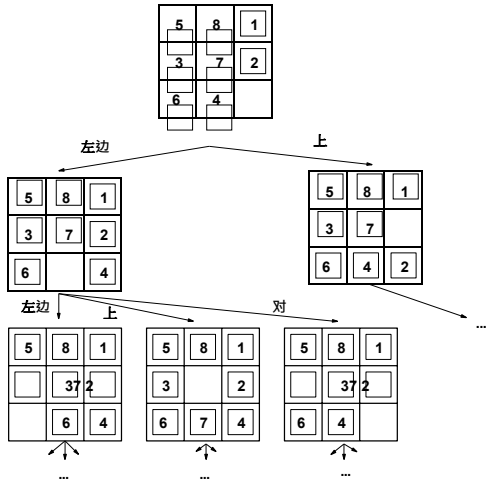
回到激励的例子

- 注意问题表述中的假设（抽象程度）。
- 请注意，虽然人们可以通过 "看
  "地图来看到解决方案，但计算机必须通过探索来构建地图
  - 从阿拉德可以去哪里？
  - 锡比乌、蒂米什瓦拉、泽林德
  - 从锡比乌可以去哪里？
- 提问的顺序决定了搜索策略
- 问题制定的假设对原始问题的解决方案的质量有很大影响

明确的状态空间

- 从寻找图的路径的角度来看状态空间搜索
- 图形$G = (V, E)$ - V: 顶点; E: 边缘
- 边缘可能有相关的成本；路径成本=路径中边缘成本之和
- 从顶点s到g的路径 - 顶点的序列$s = n_0, n_1, ..., n_k = g$
  这样，有一条边从n_i到n_{i+1}
- 状态空间图--
  节点代表状态；边代表由于行动从一个状态到另一个状态的变化；成本可能与顶点和边相关（因此是路径）。

状态空间--8人拼图



并发症

- 前向（后向）分支因子--从（到）节点的最大#出（入）弧数

- 单一状态--
  代理在已知的世界状态下开始，并知道在一个给定的行动后它将处于哪一个独特的状态。
- **多重状**态--
  对世界状态的有限访问意味着代理人不确定世界状态，但可能能够将其缩小到一组状态。
- 应变问题--
  如果代理人不知道行动的全部效果（或有其他事情发生），它可能不得不在执行过程中感知（动态改变搜索空间）。
- **探索**问题--
  不知道行动（或状态）的效果，所以代理人必须进行实验

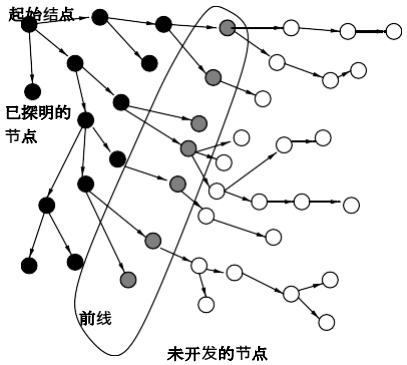
搜索方法能够处理单状态问题和多状态问题，但要付出额外的复杂性代价

新南威尔士大学
©W.Wobcke等人，2019-2022年

无信息的（盲）搜索算法

- 广度优先搜索
- 统一成本搜索
- 深度优先搜索
- 深度有限的搜索
- 迭代深化搜索
- 双向搜索

一般搜索空间（不是状态空间）



一般搜索程序

函数 GeneralSearch(problem, strategy) 返回
一个解决方案或失败，使用问题的初始状态初始化搜索图。

循环

如果没有候选扩展节点，则返回

失败，根据策略选择一个前沿节点进行扩展，如果

该节点包含一个目标状态，则返回解决方案。

否则就展开节点，并将产生的节点添加到搜索图中。

结束

注意：只在扩展节点时测试是否处于目标状态，而不是在向搜索图添加节点时测试（除了广度优先搜索！）。

状态空间与搜索空间

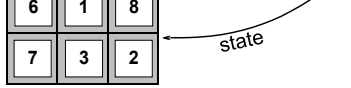
- 状态是搜索问题表述的一部分
- 节点是搜索图/树中使用的一种数据结构，包括。
  - 父，操作者，深度，路径成本 $g(x)$
- 国家没有父母，没有孩子，没有深度，也没有路径成本!

父母，行动

国家

搜索策略--边界扩展的方式

两个不同的节点可以有相同的状态



深度=6

g = 6

新南威
尔士大
学
©W.Wobcke等人，2019-
2022年

评估搜索算法

- 完整性：当存在一个解决方案时，策略保证能找到一个解决方案？
- 时间的复杂性：找到一个解决方案需要多长时间？
- 空间复杂性：搜索过程中需要的内存？
- 最优性：当存在几个解决方案时，它是否能找到 "最佳"？

注意：状态是在搜索过程中构建的，而不是事先计算出来的，所以有效地计算继任状态是至关重要的。

算法的分析--大O

- $T(n)$是$O(f(n))$意味着有一些n_0和k使得

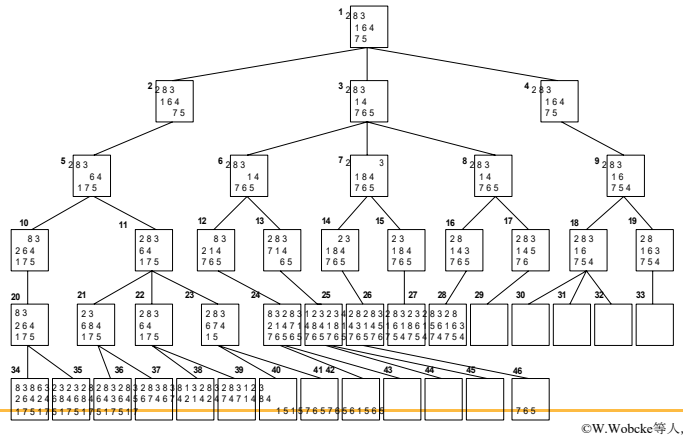
  对于每个大小为$n \geq n$的问题，$T(n) \leq k f(n)$。

- 不受实现、编译器、固定开销的影响，...
- $O()$对常数因素进行了抽象
- 实例
  - $O(n)$算法优于$O(n^2)$算法（从长远来看）。
  - 对于$n>110$，$T(100 - n + 1000)$比$T(n^2 + 1)$好。
  - 多项式$O(n^k)$比指数式$O(2^n)$好得多。

广度优先搜索

- 思路。展开根节点，然后展开根的所有子节点，再展开它们的子节点，...
- 深度为d的所有节点在$d+1$的节点之前被展开
- 可以通过使用一个队列来存储前沿节点来实现
- 广度优先搜索找到最浅的目标状态
- 当有目标状态的节点产生时停止
- 包括检查生成的状态是否已经被探索过
  - 需要一个新的数据结构，用于探索状态的集合

广度优先搜索



- $O()$符号是精确性和易于分析之间的折中。

新南威
尔士大
学
©W.Wobcke等人，2019-
2022年

广度优先搜索--分析

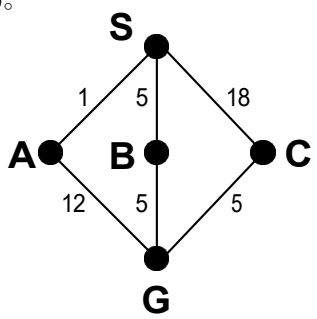
- **完整的**
- **最优 – 只要路径成本是**节点深度的非递减函数
- 产生的最大节点数：$b + b^2 + b^3 + . . + b^d$
  (其中b = 前向分支因子；d = 通往解决方案的路径长度)
- 时间和空间要求相同 $O(b^d)$

统一成本搜索

- 也被称为最低成本优先搜索
- 最浅的目标状态不一定是成本最低的解决方案
- 思路。扩大成本最低（由路径成本$g(n)$衡量）的节点
- 按照路径成本的递增顺序排列前沿的节点
- 广度优先搜索≈均匀成本搜索，其中$g(n)=$*深度*(n)
  (除了广度优先搜索在目标状态产生时停止)
- 包括检查生成的状态是否已经被探索过了
- **包括**测试，以确保边界在任何状态下只包含一个节点--
  最低成本的路径

统一费用 搜索

只有在目标节点扩展时停止，而不是在目标节点生成时停止，统一成本搜索才是最佳的。



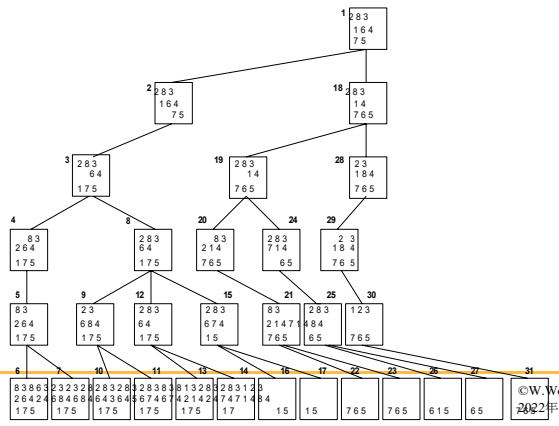
统一成本搜索--分析

- **完整的**
- 最优--只要路径成本不沿路径减少（即
  $g(succeror(n)) \geq g(n)$ for all n)
- **当路径成本是沿路**应用运算符的成本时，是合理的假设。
- 当$g(n)=depth(n)$时，表现得像广度优先搜索。
- **如果存在**负成本的路径，需要穷举搜索

深度优先搜索

- 主意。始终在树的最深层扩展节点，当搜索遇到死胡同时，再返回到较浅层的扩展节点。
- 可以使用已探索+前沿节点的堆栈来实现
- **在任何**时候，深度优先搜索都会存储从根到叶的单一路径，以及沿路节点的任何剩余未扩展的兄弟姐妹。
- 当具有目标状态的节点被扩展时停止
- **包括**检查生成的状态是否已经沿着路径被探索过--**循**环检查

深度优先搜索



深度优先搜索

```
def search(self):
    """返回从起始节点到目标节点的（下一个）路径。
    如果不存在路径，则返回无。
    """
    while not self.empty_frontier():
        path = self.frontier.pop()
        self.num_expanded += 1
        if self.problem.is_goal(path.end()): # 找到解决方
            案 self.solution = path          # 存储解决
            方案
            返回路径，
        否则。
            neighs = self.problem.neighbors(path.end() for
            arc in reversed(neighs):
                self.add_to_frontier(Path(path,arc))
    # 没有更多的解决方案
```


深度优先搜索--分析

- 存储。$O(bm)$节点(其中m=搜索树的最大深度)
- 时间。$O(b^m)$
- 在问题有很多解决方案的情况下，深度优先搜索可能优于广度优先搜索，因为它很有可能在只探索了空间的一小部分后找到一个解决方案。
- 然而，即使在相对较浅的层次上存在解决方案，深度优先搜索也可能卡在深层或无限的路径上。
- **因此，深度**优先搜索是不完整的，也不是最优的

- 对于有深度或无限路径的问题，避免深度优先搜索

新南威
尔士大
学
©W.Wobcke等人，2019-
2022年

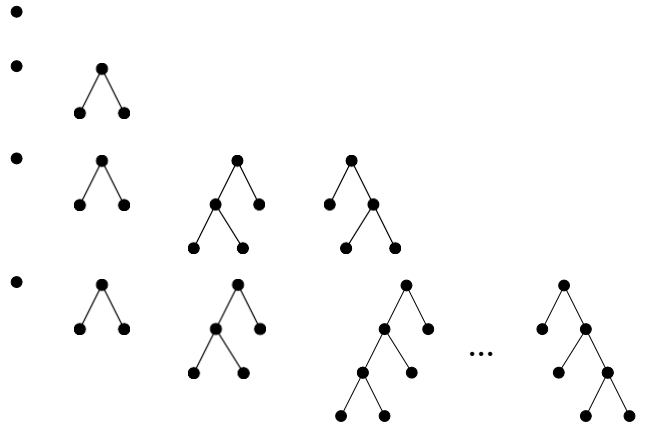
深度有限的搜索

- 想法：对路径的深度施加约束
- **在一些**问题中，你可能知道应该在一定的成本内（例如一定的步数）找到一个解决方案，因此没有必要在这一点之外搜索解决方案的路径。
- **分析**报告
  - 完整但不是最优（可能找不到最短的解决方案）
  - 然而，如果选择的深度限制太小，可能找不到解决方案，在这种情况下，深度限制的搜索是不完整的
  - 时间和空间复杂性类似于深度优先搜索（但相对于深度限制而不是最大深度）。

迭代深化搜索

- 要决定搜索的深度限制可能非常困难
- 任何两个节点之间的最大路径成本被称为状态空间的直径
- 这将是一个很好的深度限制的候选者，但可能很难事先确定。
- 主意。依次尝试所有可能的深度限制
- 结合了深度优先和广度优先搜索的优点

迭代深化搜索



迭代深化搜索--分析

- 最优；完整；空间 - $O(bd)$
- 一些州被多次扩大。这不是浪费吗？
  - 扩展到深度d的次数=$1 + b + b^2 + b^3 + . . + b^d$
  - 因此，对于迭代深化，总扩展量=（d+1）1+（d）b+（d-1）$b^2 + . . + 3b^{d-2} + 2b^{d-1} + b^d$

  分支系数越高，开销就越低（即使对于b=2，搜索所需时间约为两倍）
  - 因此时间复杂度仍为$O(b^d)$

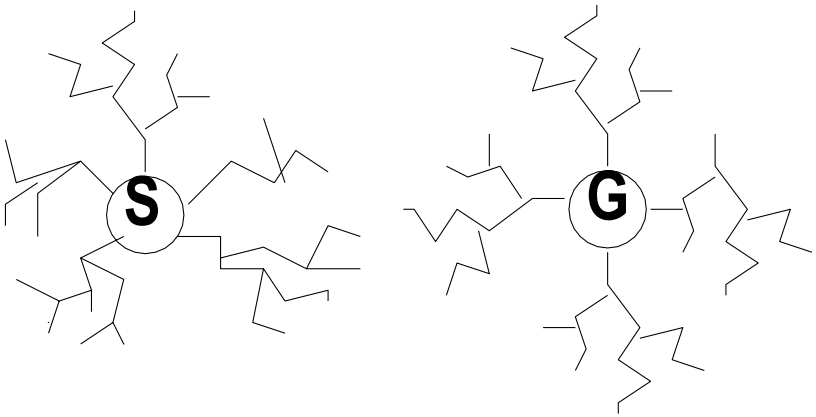
- 可以在每次迭代时将深度限制增加一倍 - 开销为$O(dlogd)$。
- **一般来**说，对于不知道解的深度的大搜索空间，迭代深化是首选的搜索策略。

新南威尔士大学
©W.Wobcke等人，2019-2022年

双向搜索

- 思路。同时从初始状态向前搜索和从目标状态向后搜索，直到两者相遇。
- 为了向后搜索，我们需要生成状态的前身（这并不总是可能或容易的）。
- 如果运算符是可逆的，则继承集和前任集是相同的
- **如果有很多目**标状态，也许多状态搜索会起作用（但不是在国际象棋中）。
- 需要检查一个节点是否同时出现在两个搜索中--效率可能很低
- 每一半的最佳搜索策略是什么？

双向搜索



双向搜索--分析

- 如果在深度d存在解决方案，那么双向搜索需要时间 $O(2b^{\frac{d}{2}}) = O(b^{\frac{d}{2}})$（假设交叉点的检查时间恒定）。
- 为了检查交叉点，必须在内存中拥有其中一个搜索的所有状态，因此空间复杂度为$O(b^{\frac{d}{2}})$

摘要 - 盲目搜索

准则	广度首先	统一的费用	深度-首先	深度-ĀĀĀ	迭代式深化	双向的
时间	bd	bd	bm	*盲文*	bd	$b^{\frac{d}{2}}$
空间	bd	bd	bm	*盲文*	bd	$b^{\frac{d}{2}}$
最优	是	是	没有	没有	是	是
完整的	是	是	没有	是的，如果$l \geq d$	是	是

b - 分支因子

d - 最浅的解决方案的深度

m - 树的最大深度

l - 深度限制

新南威
尔士大
学

©W.Wobcke等人，2019-
2022年