

A* 搜索



A* 搜索



基于成本的统一搜索法 (UCS)



贪婪法 (Greedy)

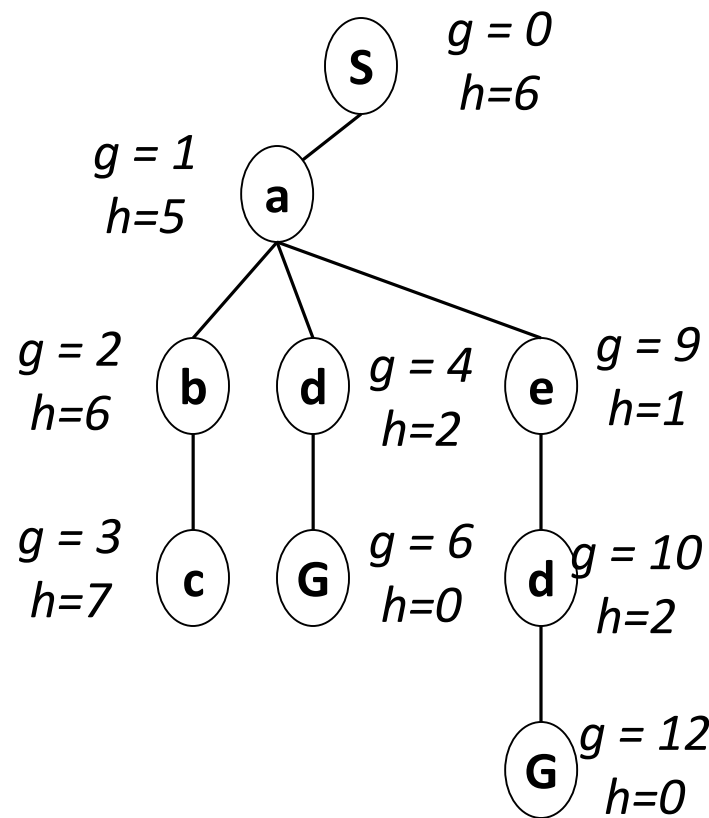
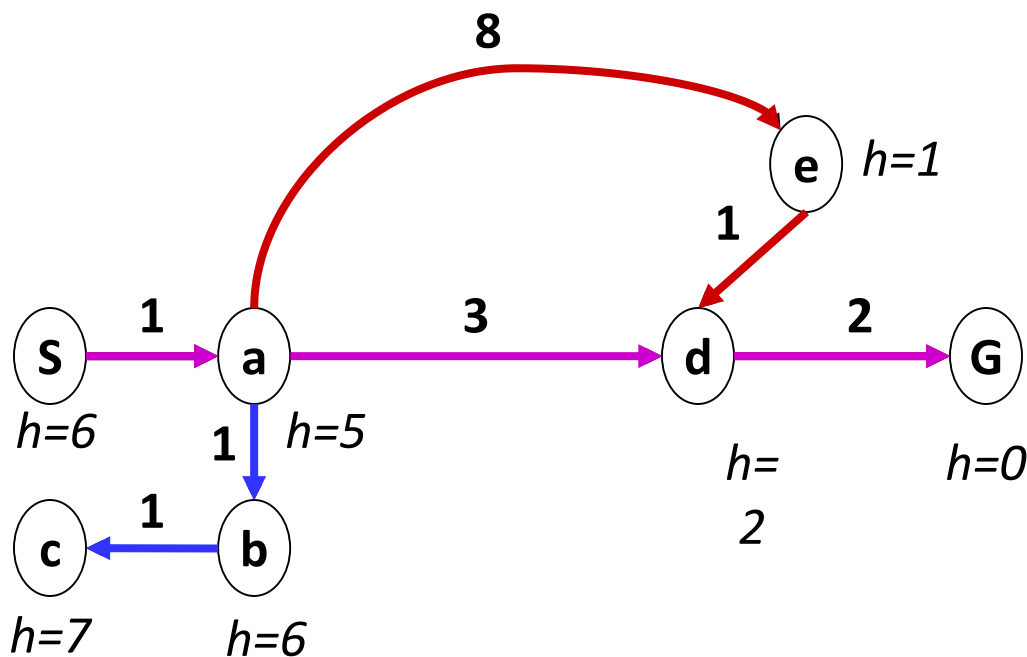


A*

结合统一搜索和贪婪搜索

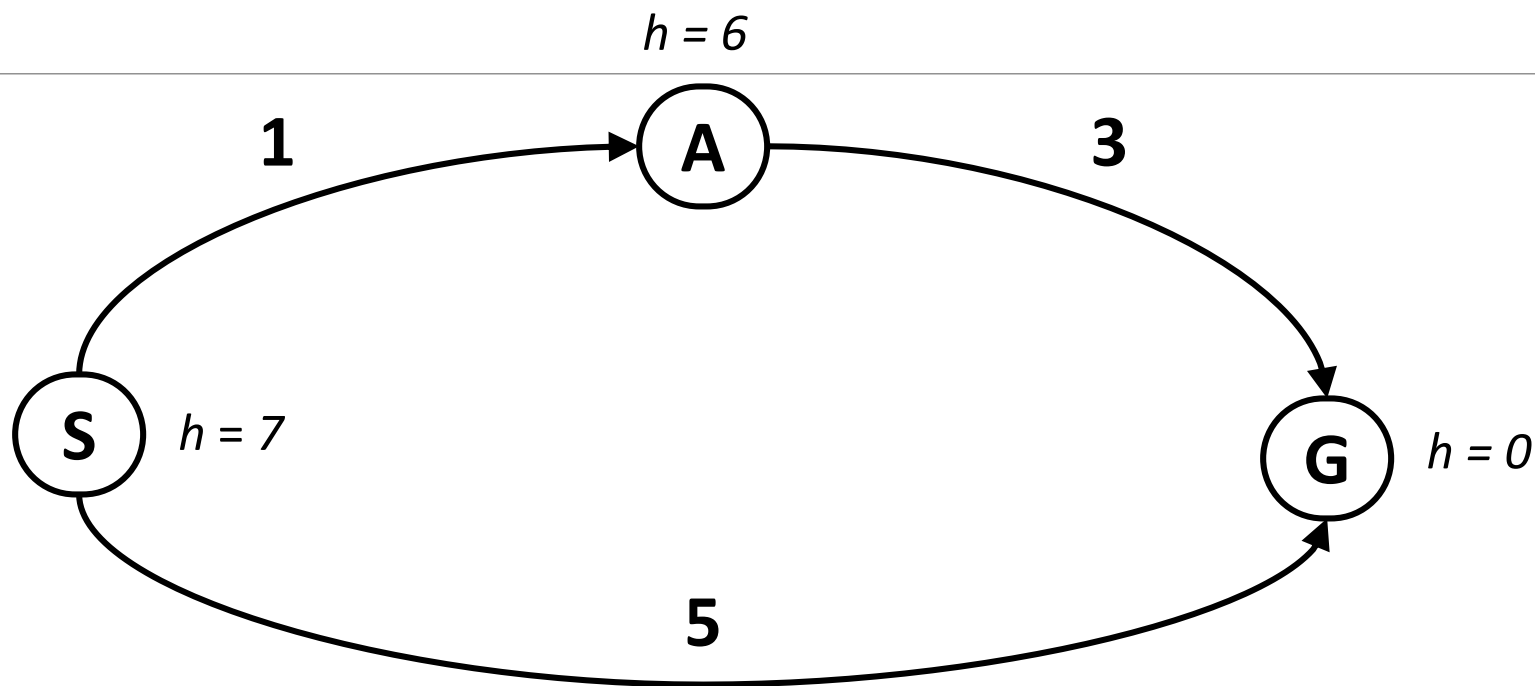
统一：把路径成本排序，即来程的成本 $g(n)$

贪婪：排序按照与目标的临近性，即前程成本 $h(n)$



A* 搜索 : $f(n) = g(n) + h(n)$

A* 是最优的吗？

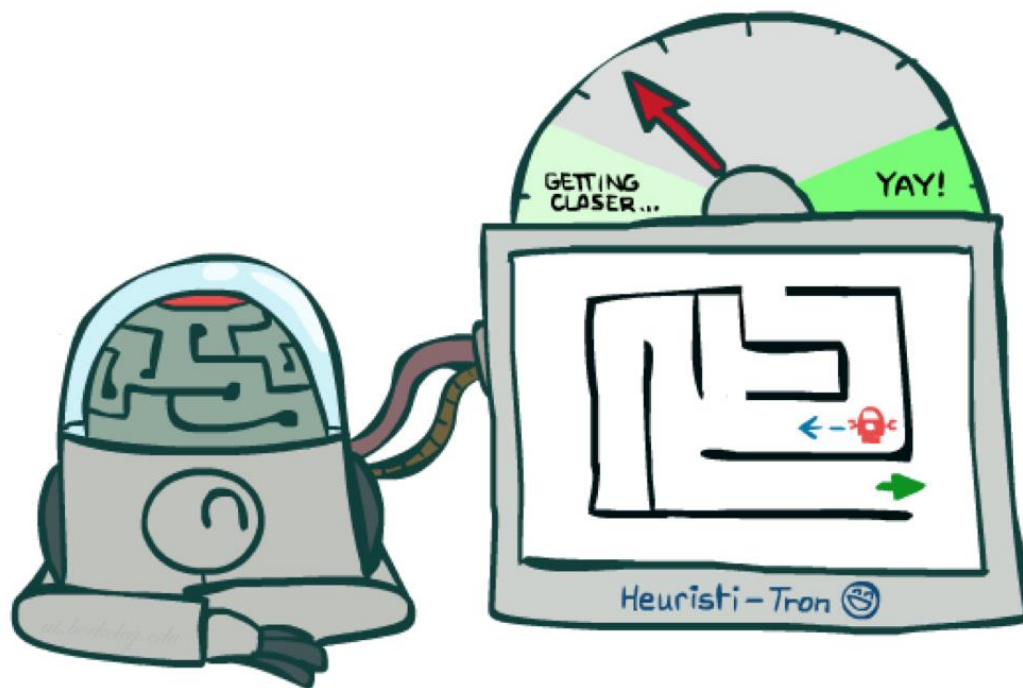


哪个地方错了？

实际到目标成本 < **估计**的到目标成本

我们需要估计值小于实际成本

可接纳的启发式函数



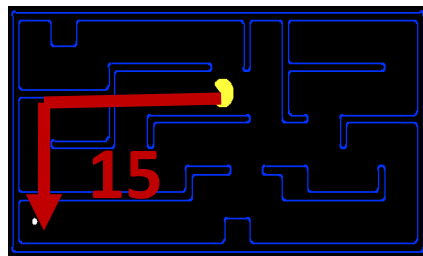
可接纳的启发式函数 (Admissible Heuristics)

h 是 *可接纳的* (乐观的, 想的比实际好):

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

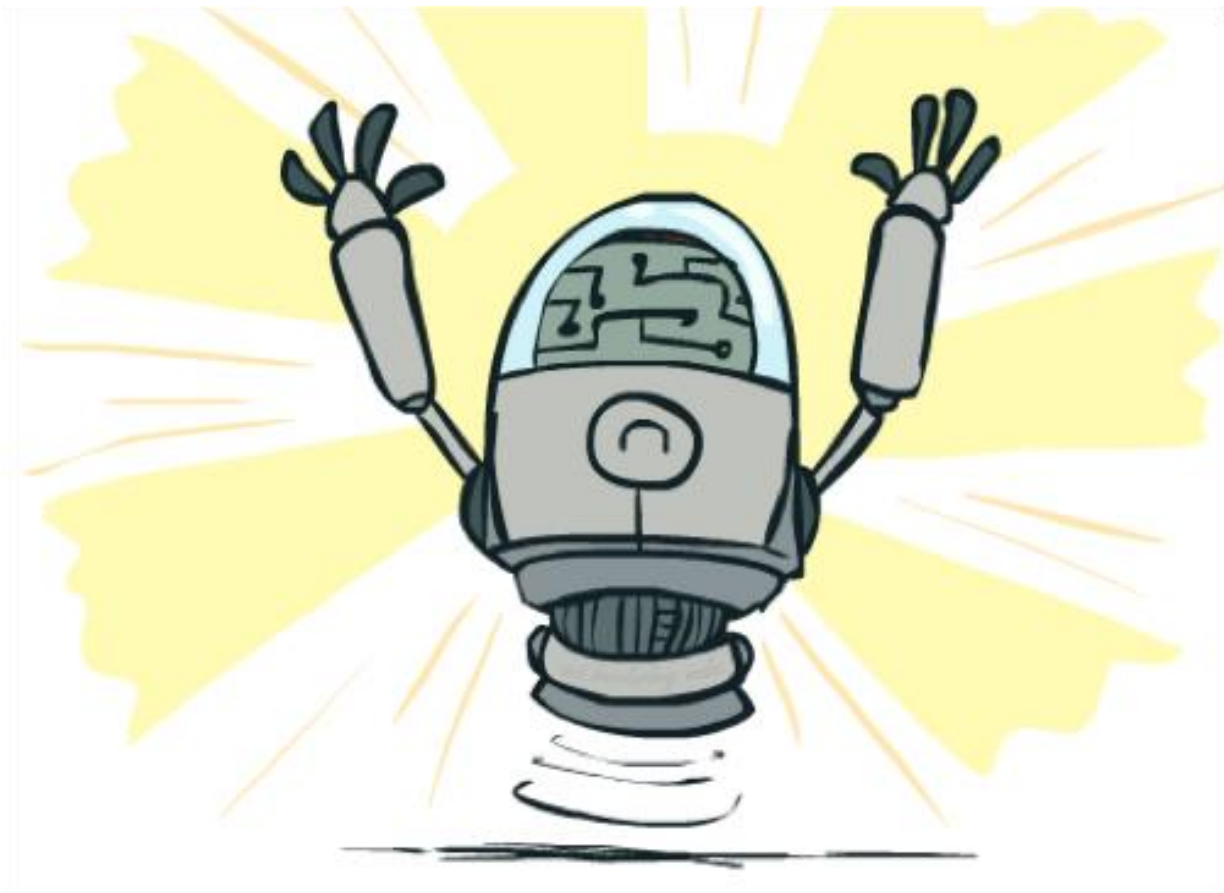
$h^*(n)$ 是到一个最近的目标节点的真成本值

举例:



在实际应用A*算法时, 一个主要任务就是设计可接纳的启发式函数。

A* 树搜索的最优性



A* 树搜索的最优性

假定:

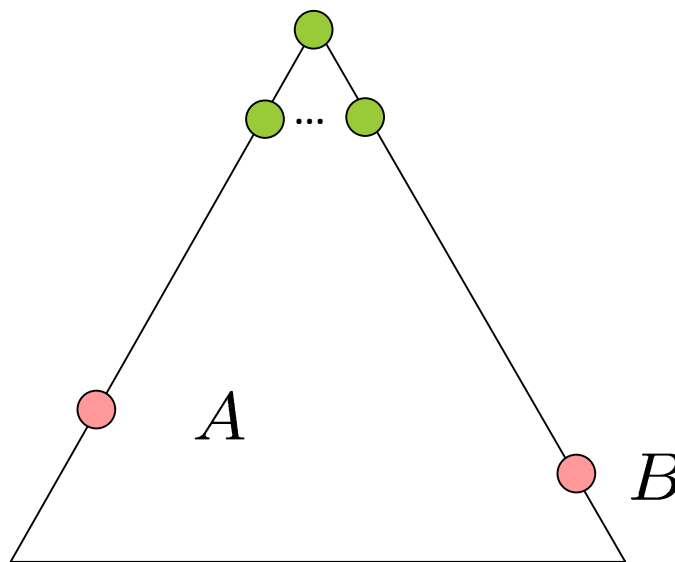
A : 一个最优目标节点

B : 一个次优目标节点

h 是可接纳的

如果最优性成立, 那么:

A 将会在**B**之前被选择先扩展



A* 树搜索的最优性

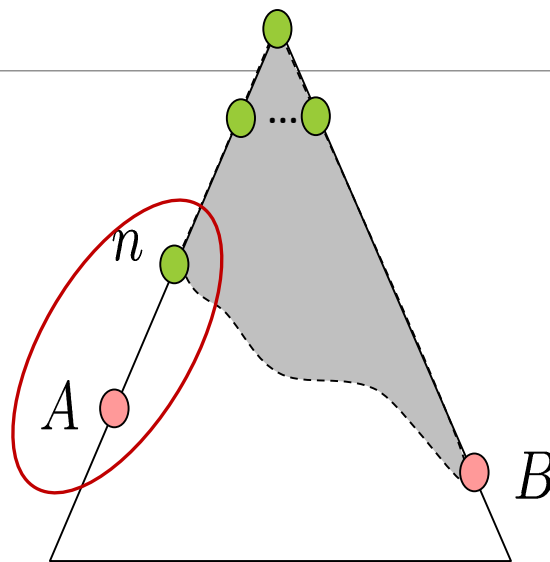
证明:

假设 **B** 在搜索前沿里

A 的某个祖先节点 **n** 也在搜索前沿里

宣称: **n** 会先于 **B** 被选择扩展

1. $f(n)$ 小于或等于 $f(A)$



$f(n) = g(n) + h(n)$ f 的定义

$f(n) \leq g(A)$

$g(A) = f(A)$

h 的可接纳性

$h = 0$ 当在目标节点

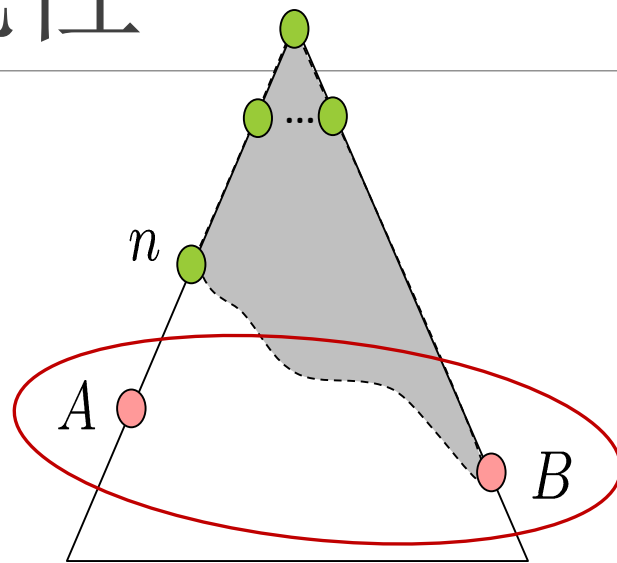
证明: A^* 树搜索的最优性

假设 B 在搜索前沿里

A 的某个祖先节点 n 也在搜索前沿里

宣称: n 会先于 B 被选择扩展

1. $f(n)$ 小于或等于 $f(A)$
2. $f(A)$ 小于 $f(B)$



$$g(A) < g(B)$$

$$f(A) < f(B)$$

B 次优解

$h = 0$ 在目标节点

A* 树搜索的最优性

证明:

假设 **B** 在搜索前沿里

A 的某个祖先节点 **n** 也在搜索前沿里

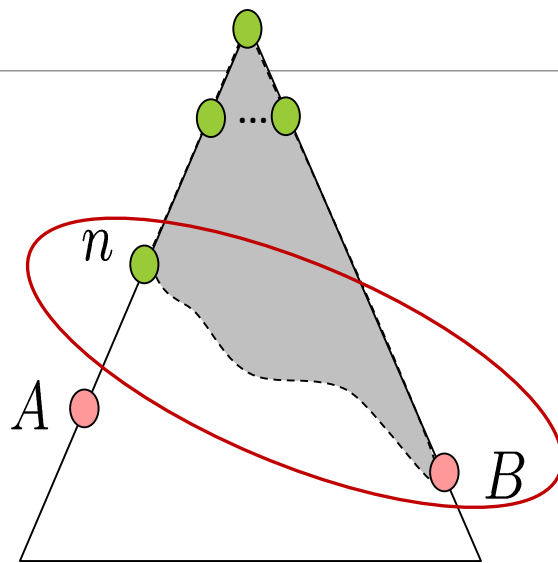
宣称: **n** 会先于 **B** 被选择扩展

1. $f(n)$ 小于或等于 $f(A)$
2. $f(A)$ 小于 $f(B)$
3. **n** 先于 **B** 被扩展

所有 **A** 的祖先节点都会在 **B** 之前被扩展

A 先于 **B** 被扩展

A* 搜索是最优的

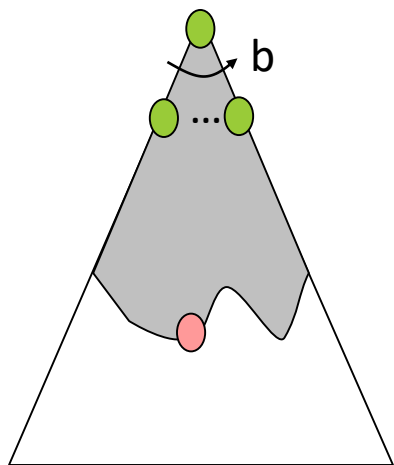


$$f(n) \leq f(A) < f(B)$$

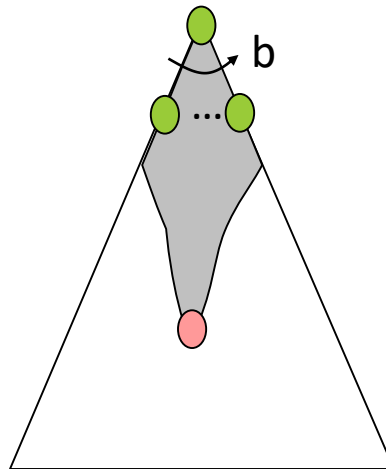
A^* 搜索的属性

A*搜索的属性

成本统一搜索

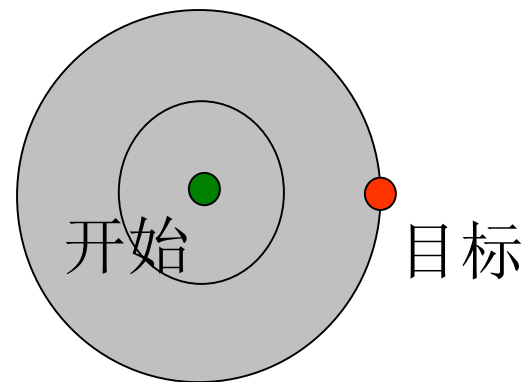


A*搜索

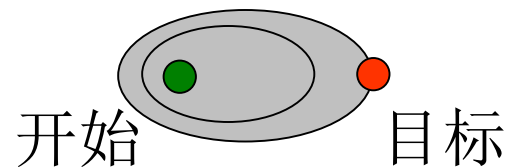


成本统一搜索 vs A* 搜索轮廓

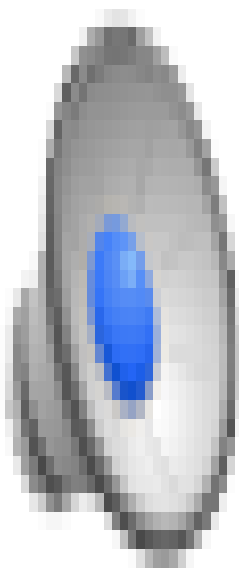
基于成本的统一搜索在各个方向上均匀探索



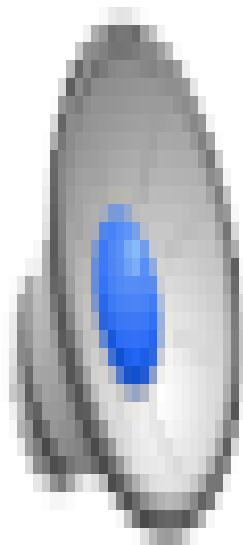
A* 在朝向目标的方向上进行探索，同时保证解的最优性



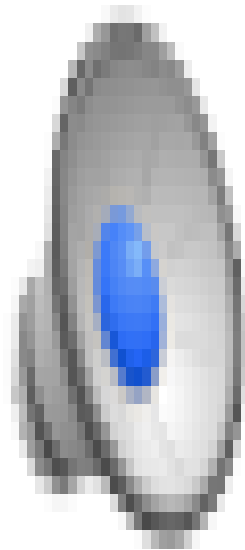
视频展示：搜索轮廓 (空迷宫) – 基于成本的统一搜索



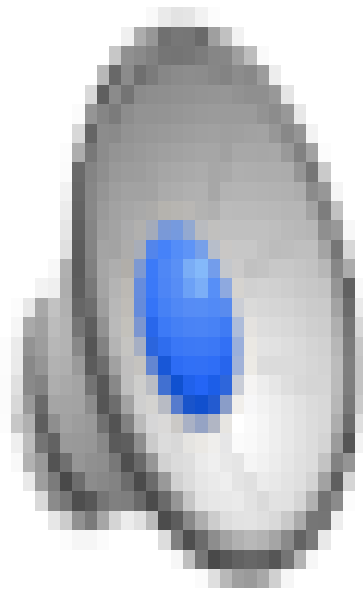
视频展示：搜索轮廓 (空迷宫) – 贪婪搜索



视频展示：搜索轮廓 (空迷宫) - A^*



视频展示：搜索轮廓 (Pacman 迷宫) - A^*



比较

贪婪搜索



A*



基于成本的统一搜索

A* 应用

视频游戏

路径搜索问题

资源规划问题

机器人移动规划

语言分析

机器翻译

语音识别

...



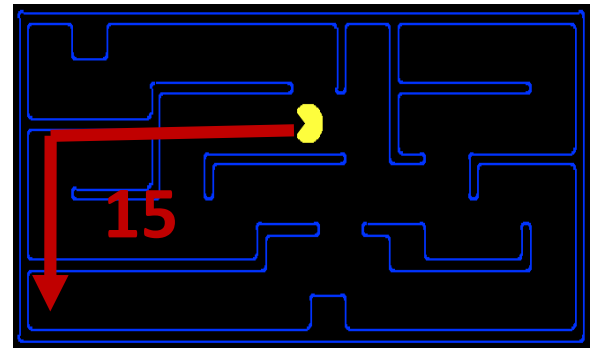
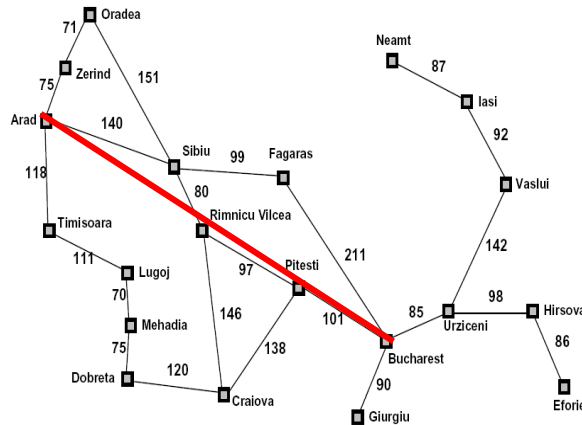
创建启发式函数

创建可接纳的启发式函数

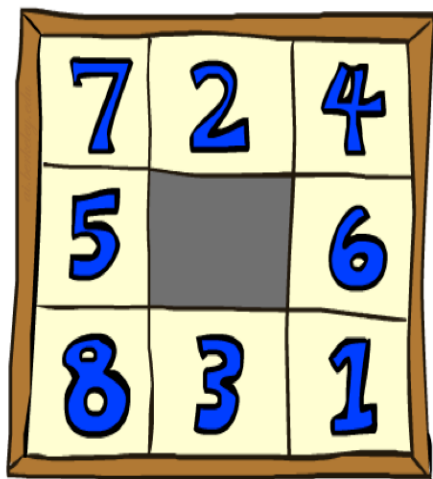
在求解很难的搜索问题时，大部分的工作是找到可接纳的启发式函数。

可接纳的启发式函数信息，通常是对应的**松弛问题(relaxed problems)**的解，解除对行动的限制。

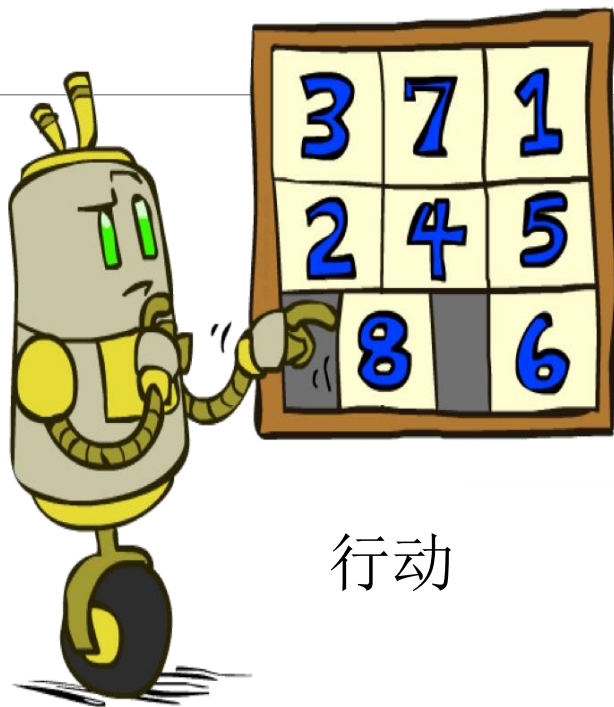
366



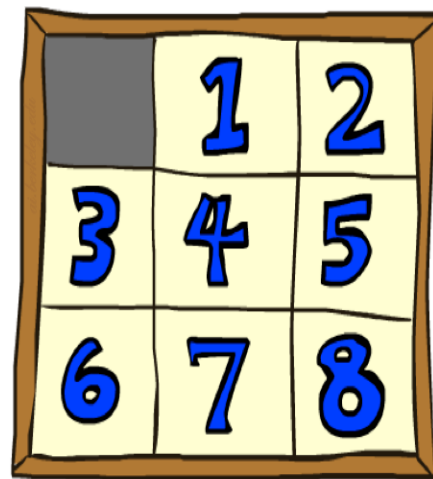
举例: 8 数字谜题



开始状态



行动



目标状态

状态? 总共有多少种状态?

行动? 开始状态有多少种可选行动?

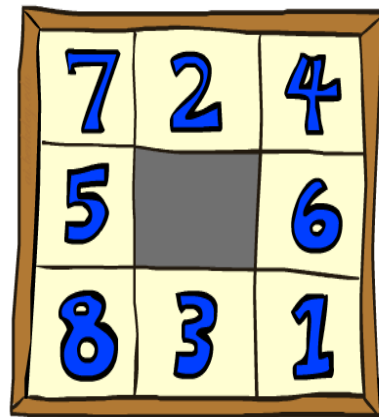
行动成本?

8 数字谜题

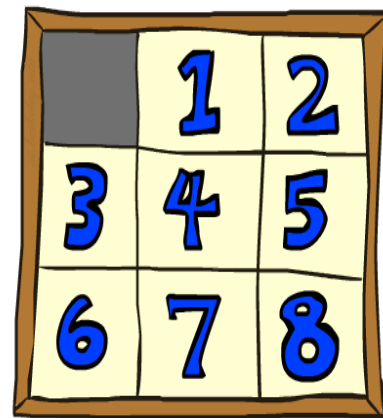
启发式: 错位方块的数量
为什么是可接纳性的?

$h(\text{开始}) = 8$

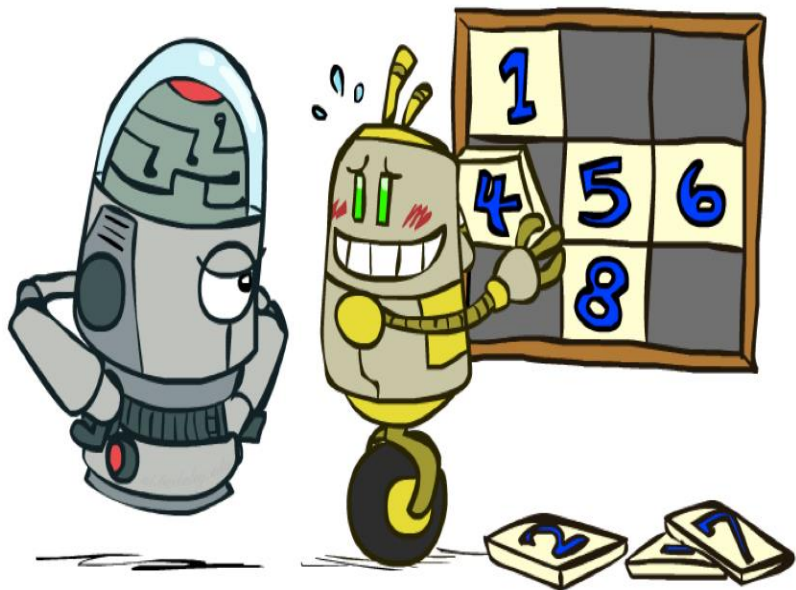
松弛问题的启发信息 (假设可直接跳到位)



开始



目标



平均节点扩展数, 当最优解的深度是:

	...4 步	...8	...12
UCS	112	6,300	3.6×10^6
A*TILES	13	39	227

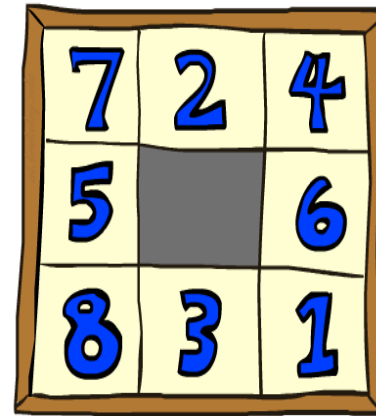
8 数字谜题， 继续

如果把条件再松弛一下，任何方块可以在任何时候朝任何方向滑动，无论那里有没有其他方块

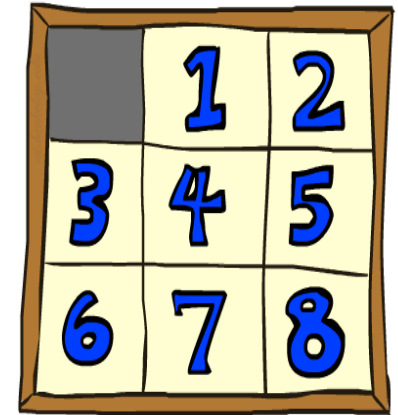
即曼哈顿距离

可接纳性的？

$$H(\text{开始}) = 3 + 1 + 2 + \dots = 18$$



开始



目标

Average nodes expanded when the optimal path has...			
	...4 steps	...8 steps	...12 steps
A*TILES	13	39	227
A*MANHATTAN	12	25	73

组合启发式（函数）信息

支配优势: $h_a \geq h_c$ 如果满足:

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

- 一般讲, 越大越好, 只要保持可接纳性。
- H 是 0 的话, 比较糟, (A^* 变成什么, 如果 $h=0$?)。
- 和真实目标成本相同最好, 但很难达到!

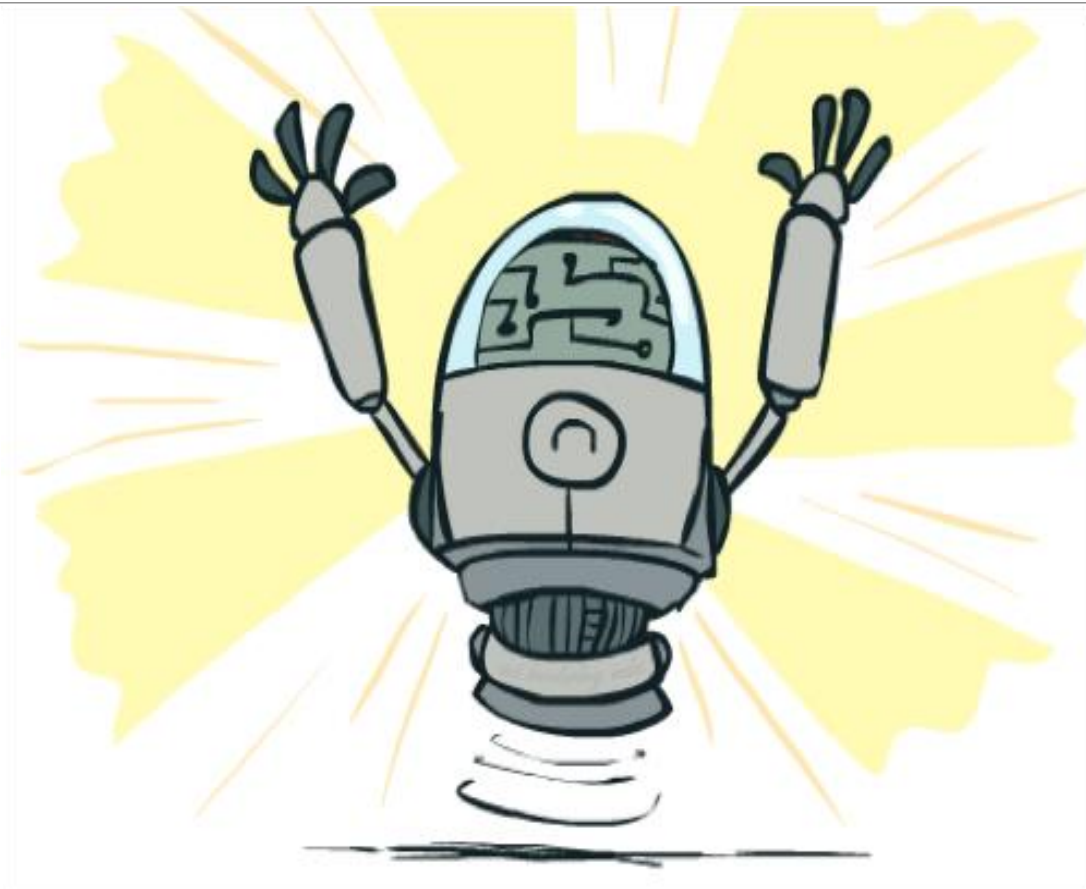
如果两个启发式函数, 都不支配对方怎么办?

- 形成一个新的, 通过最大式组合:

$$h(n) = \max(h_a(n), h_b(n))$$

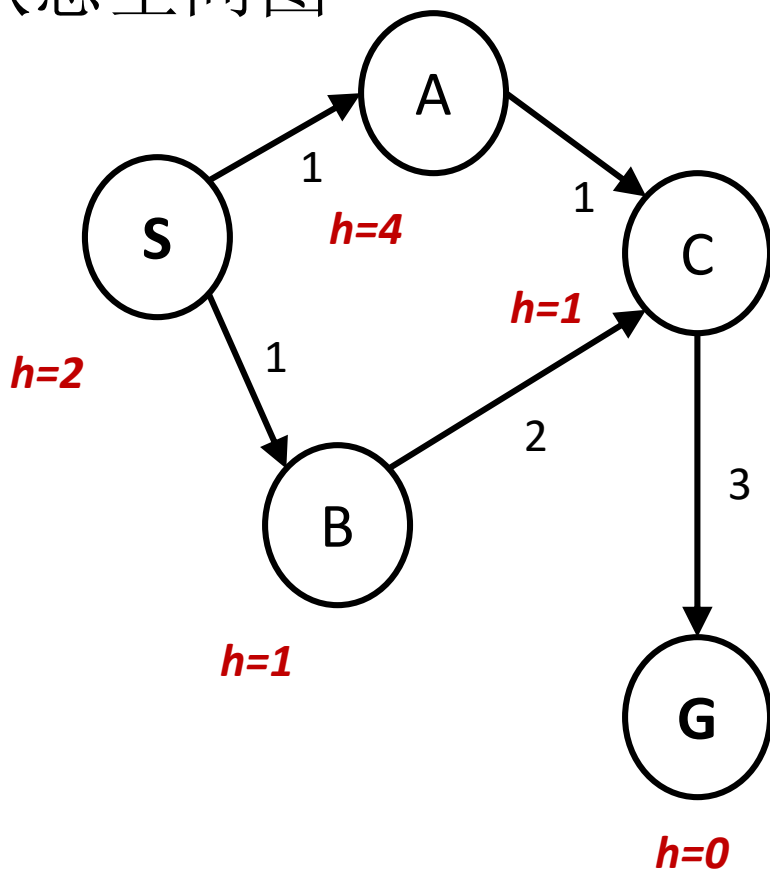
- 这个既是可接纳的, 也是对之前任一个都有支配优势的, 启发式函数。

A* 图搜索算法的 优化性

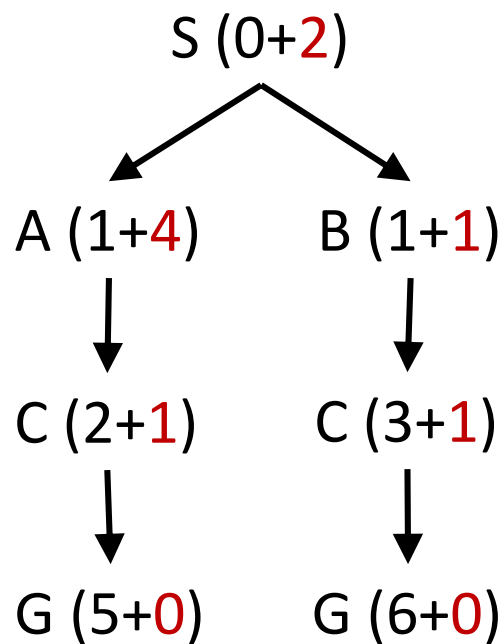


A*图搜索走错了吗？

状态空间图



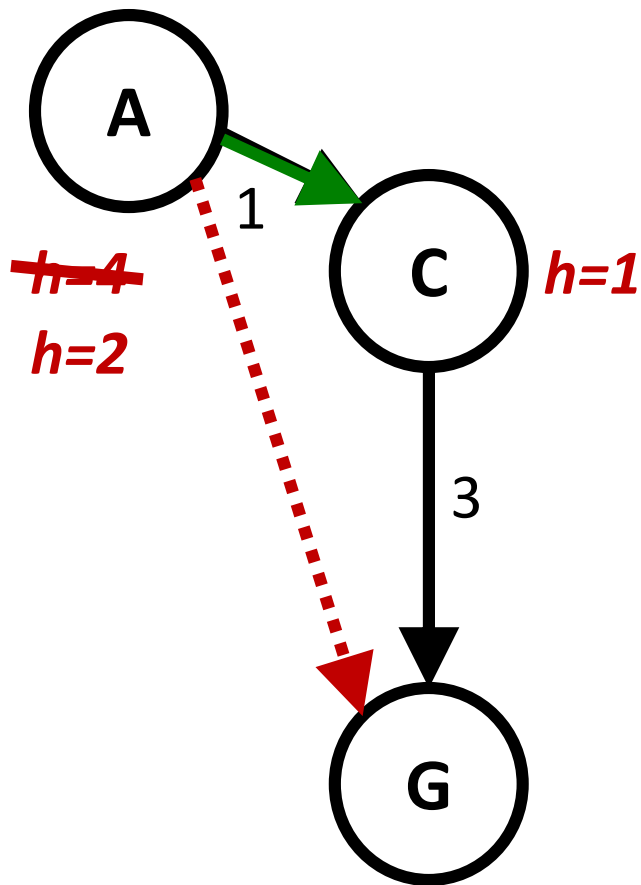
搜索树



C-G 不会被扩展，因为C 已被访问过，因此错过了最优路径。

启发性函数的一致性

主体思想: 估计成本 \leq 实际成本



- 可接纳性: 启发函数估计成本 \leq 实际路径成本

$$h(A) \leq \text{从 A 到 G 的实际路径成本}$$

- 一致性: 启发函数估计的 **步骤成本** (弧成本) \leq 实际步骤成本

$$h(A) - h(C) \leq \text{cost(A 到 C)}$$

一致性的结果:

- f 值 在同一路径搜索中不会减少

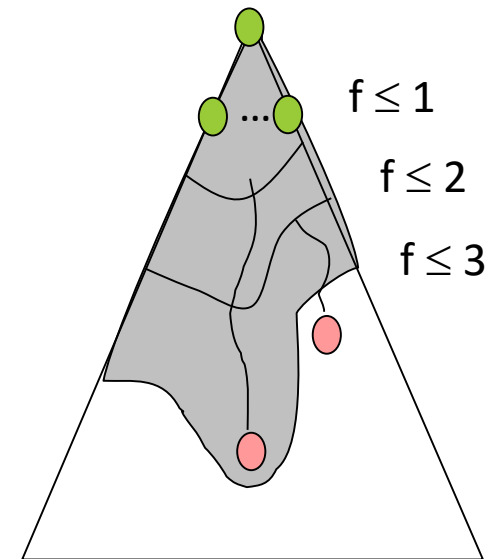
- $h(A) \leq \text{cost(A 到 C)} + h(C)$

- A* **图搜索**是最优的 (找到的解是最优的)

A* 图搜索的最优性 (Optimality)

如果 A* 和一个一致性的启发式函数，那么：

- 事实 1: A* 扩展节点过程中， f 值递增 (f -轮廓)
- 事实 2: 对于每个状态 s , 到达 s 的最优路径上的节点先于到达 s 的次优路径节点被扩展
- 结果: A* 图搜索是最优的



最优性

树搜索:

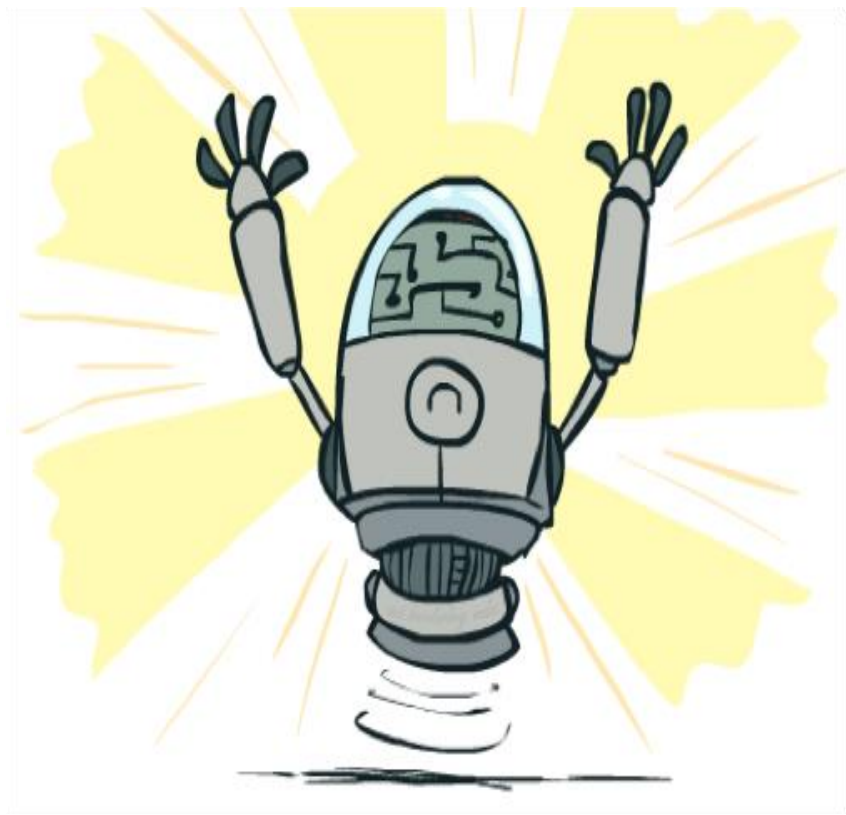
- A* 最优, 如果启发式函数是满足可接纳性的
- 基于成本的统一搜索(UCS) 是一个特例 ($h = 0$)

图搜索:

- A* 最优, 如果启发式函数是满足一致性的
- UCS 最优 ($h = 0$ 也是一致性的)

一致性蕴涵了可接纳性

通常, 从条件松弛问题中得到的启发式信息趋向于满足一致性条件。

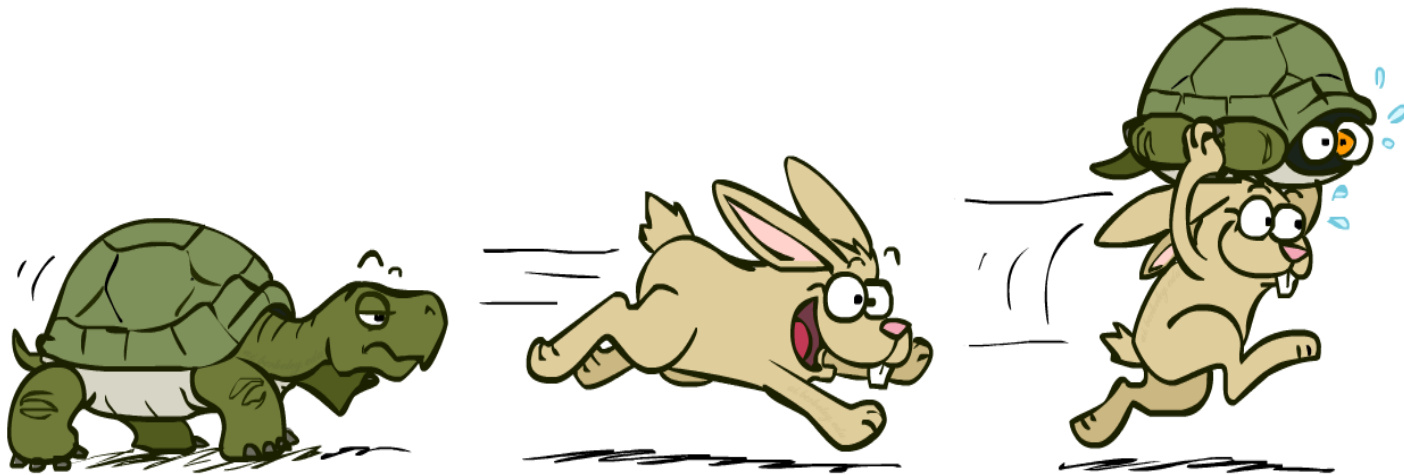


A*: 总结



A*: 总结

- A* 既使用了来程路径成本，又利用了估计的前程路径成本
- A* 是最优的，如果伴随使用可接纳性的或一致性的启发式函数
- 启发式函数的设计是关键: 通常运用条件松弛问题的解



AI 新闻

Dell Inc Announces \$125B Investment In China, Including Artificial Intelligence Lab

<http://www.ibtimes.com/dell-inc-announces-125b-investment-china-including-artificial-intelligence-lab-2090481>

Apple on hiring spree for AI experts

<http://www.bizjournals.com/sanjose/news/2015/09/08/apple-on-hiring-sprees-for-ai-experts.html>

Toyota Invests \$50 Million In Artificial Intelligence Research For Vehicle Robotics

<http://www.forbes.com/sites/dougnewcomb/2015/09/09/toyota-invests-50-million-in-artificial-intelligence-research-for-vehicle-robotics/>

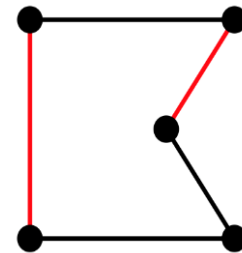
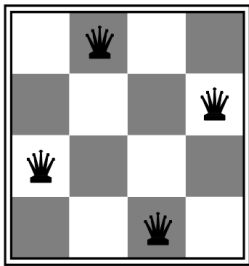
人工智能导论：局部 搜索和不确定搜索

局部搜索

许多优化问题, **路径解** 不相关; 目标状态才 **是** 问题的解

状态空间 = 完整布局配置的集合

找到 **满足约束的布局配置解**, 例如n个皇后问题; 或是找到 **最优布局解**, 例如旅行销售商问题



在这些情况, 可以使用迭代改进算法; 保持一个当前状态, 然后尝试改进它。

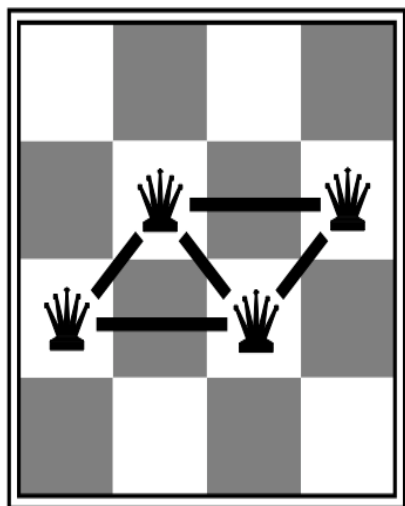
空间成本固定, 适合线上(online)或线下(offline)搜索

*N*皇后问题的启发式信息

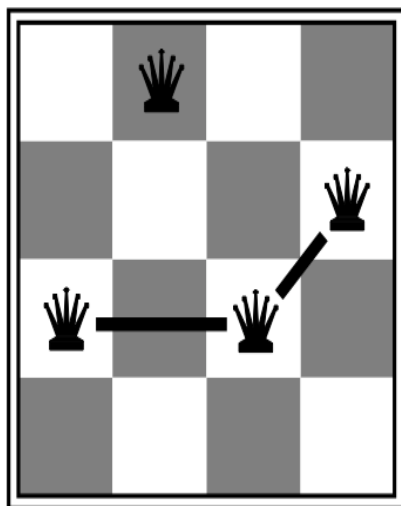
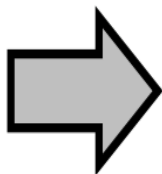
目标布局: n 个皇后在棋盘上不互相冲突

状态: n 个皇后在棋盘上布局, 一列放一个

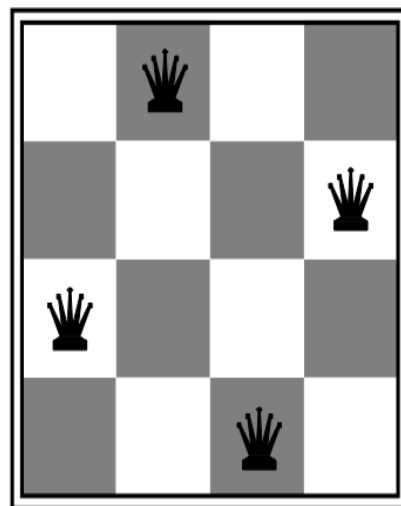
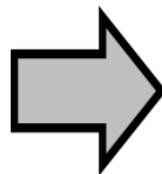
启发式函数: 相互冲突的皇后对数



$h = 5$

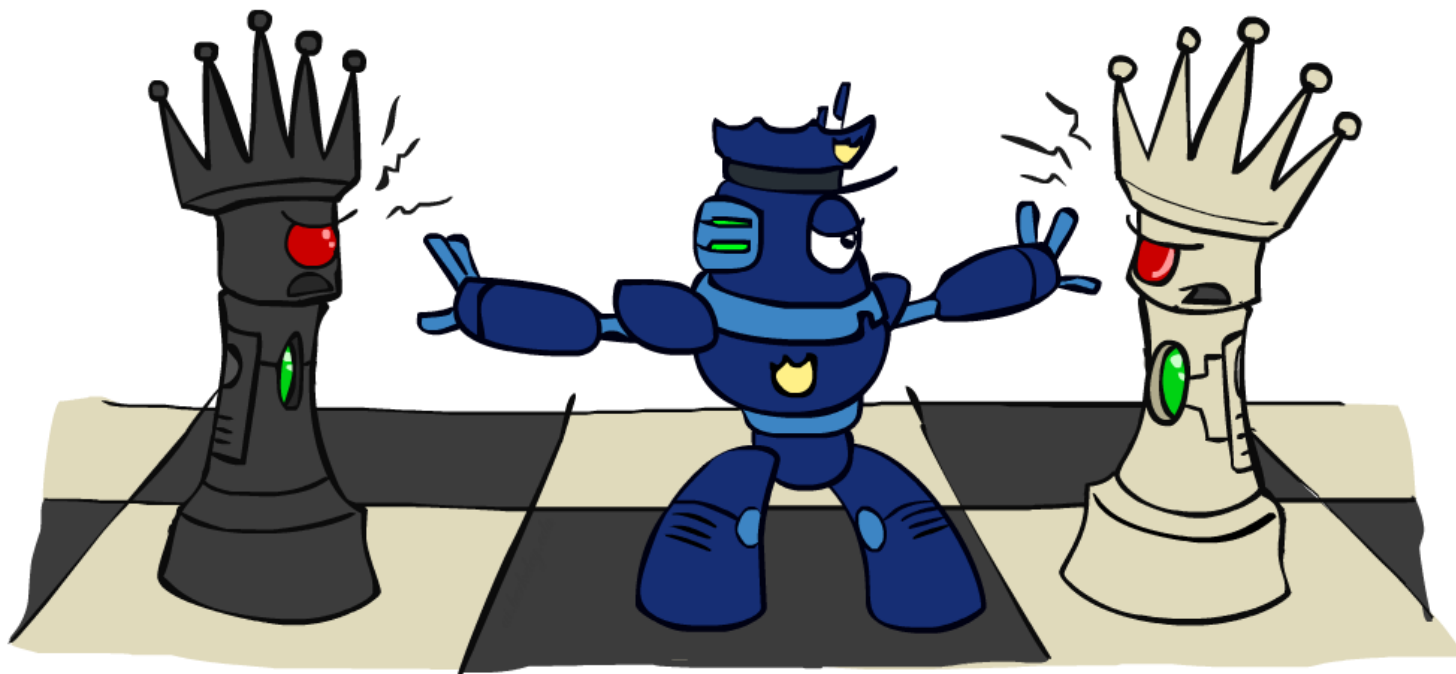


$h = 2$

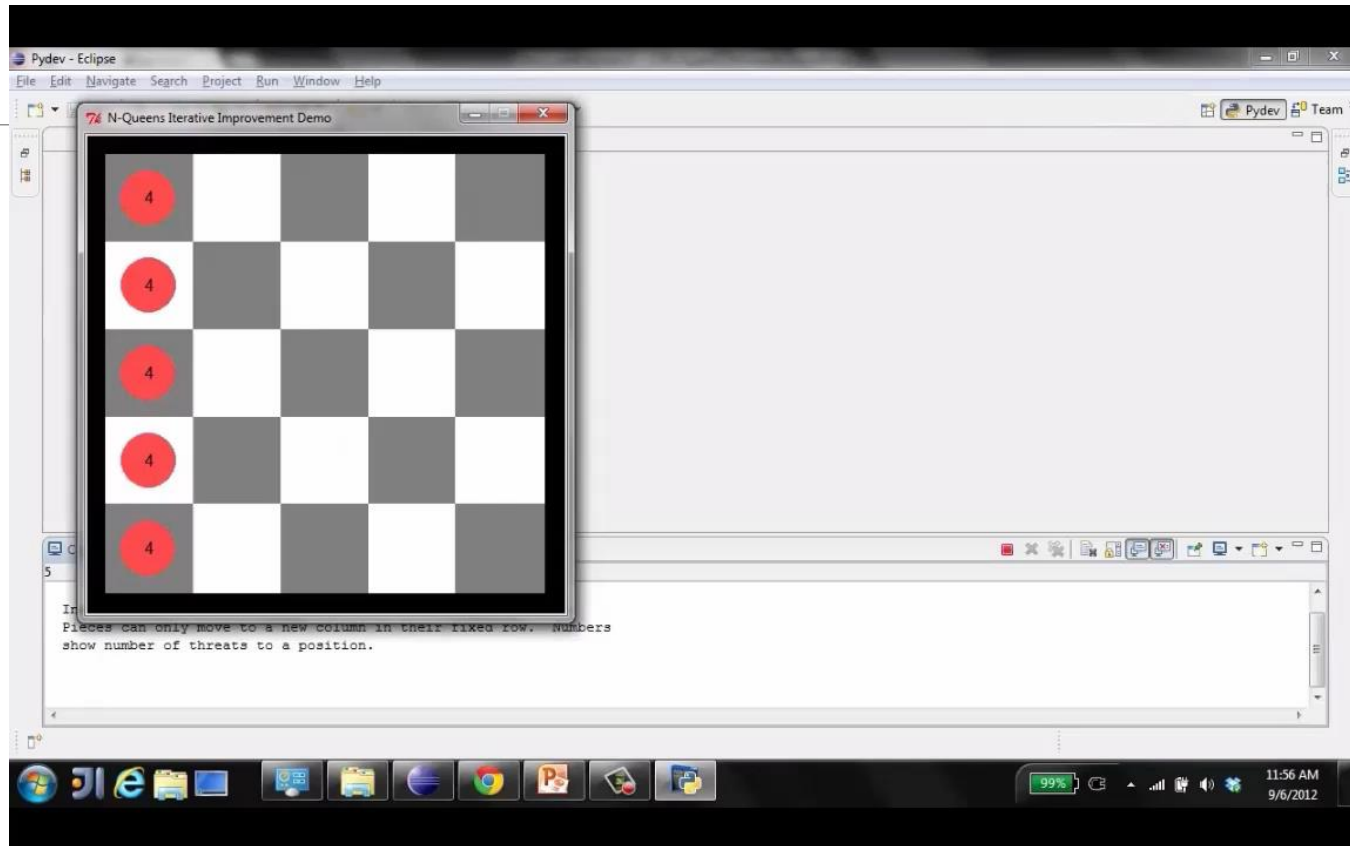


$h = 0$

N皇后演示



[Demo: n-queens – iterative improvement
(L5D1)]



爬山算法(Hill-climbing)

function HILL-CLIMBING(问题) **returns** 一个状态

当前节点 \leftarrow make-node(问题.初始状态)

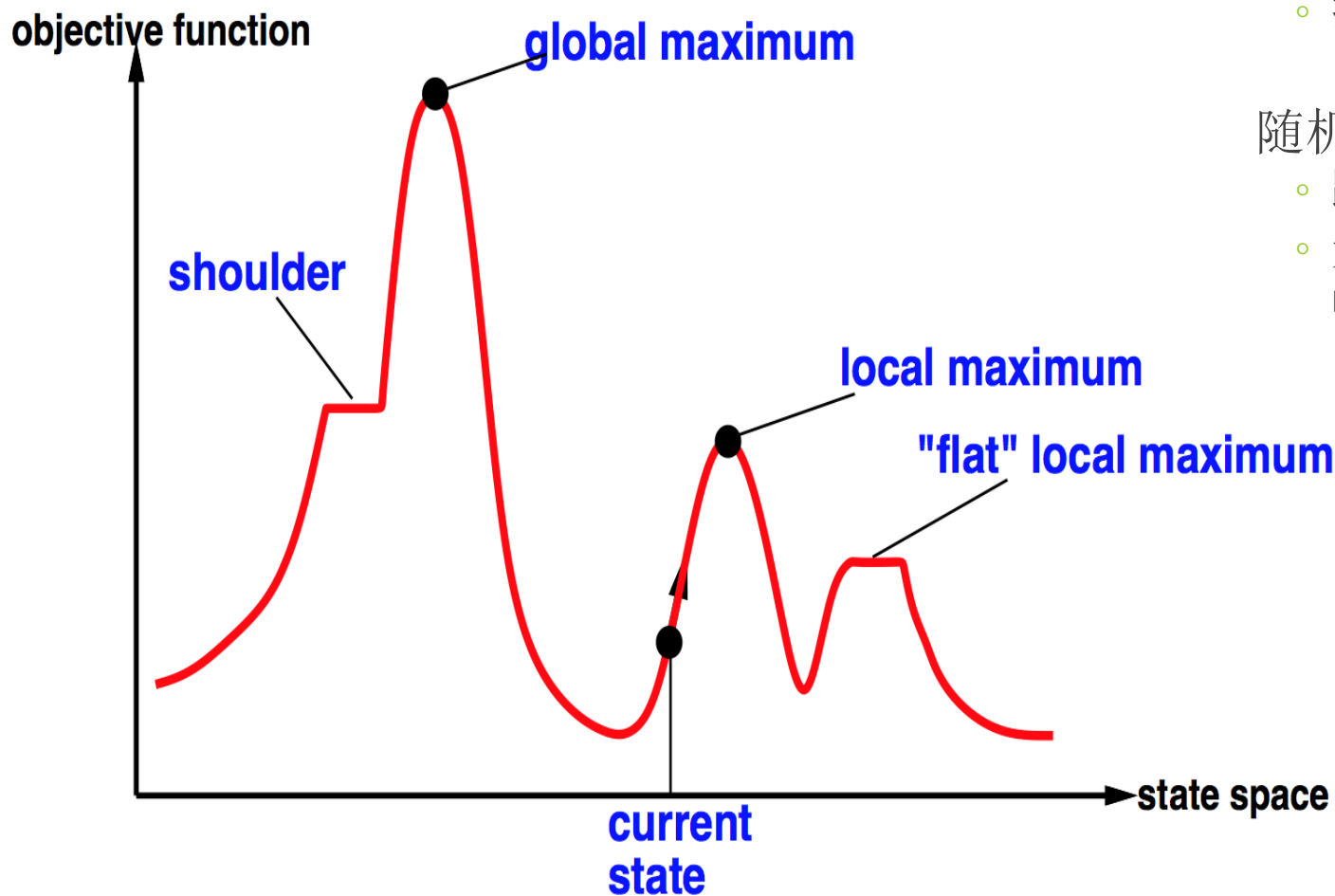
loop do

邻居节点 \leftarrow 选一个 当前节点 的后继节点中评估值最大的节点

if 邻居节点.value \leq 当前节点.value **then return** 当前节点.state

当前节点 \leftarrow 邻居节点

全局和局部最优解



随机开始点

- 找到全局最优

随机水平移动

- 跳出 “shoulder”
- 无限循环在 “flat local maxima”

模拟退火(Simulated annealing)

退火过程用来缓慢冷却金属使之达到一个稳定状态

基本想法:

- 允许偶尔的随机移动, 依赖于“温度”
- 高温 => 更多的随机移动, 系统可能走出局部最优格局
- 逐渐降低温度, 根据一个冷却的时间调度

好似晃动崎岖不平的表面, 让一个乒乓球滚入最深的曲面里

- 理论上: 存在一个冷却时间调度, 使得找到全局最优的可能概率为1。

模拟退火(Simulated annealing)算法

function SIMULATED-ANNEALING(**问题**,冷却**调度**) **returns** 一个状态

当前节点 \leftarrow make-node(**问题**.initial-state)

for $t = 1$ **to** ∞ **do**

$T \leftarrow$ 调度(t)

if $T = 0$ **then return** **当前节点**

下一节点 \leftarrow 一个随机选择的 **当前节点** 的后继节点

$\Delta E \leftarrow$ **下一节点**.value - **当前节点**.value

if $\Delta E > 0$ **then** **当前节点** \leftarrow **下一节点**

else **当前节点** \leftarrow **下一节点**, 如果随机概率大于等于 $e^{\Delta E/T}$

局部光束搜索(Local beam search)

基本思想:

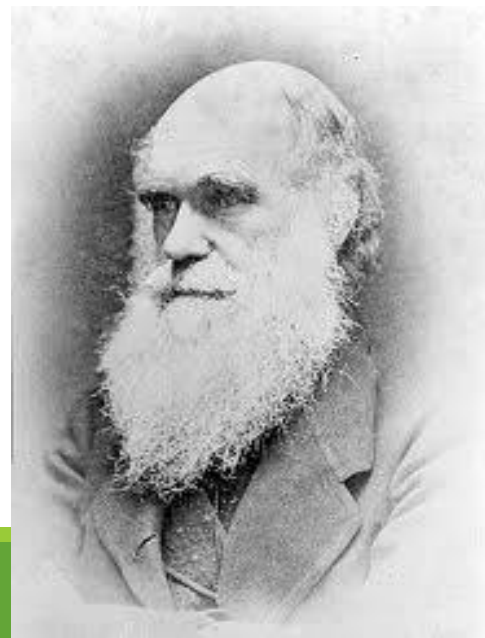
- K 拷贝某种局部搜索算法, 随机初始化
- 在每一轮
 - 从 k 个当前状态产生所有的后继状态
 - 选出 k 个最好的, 赋给当前搜索状态

为什么和 开始 K 个 并行 局部搜索不一样?

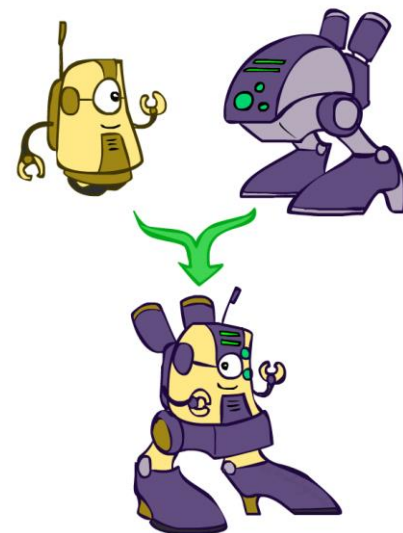
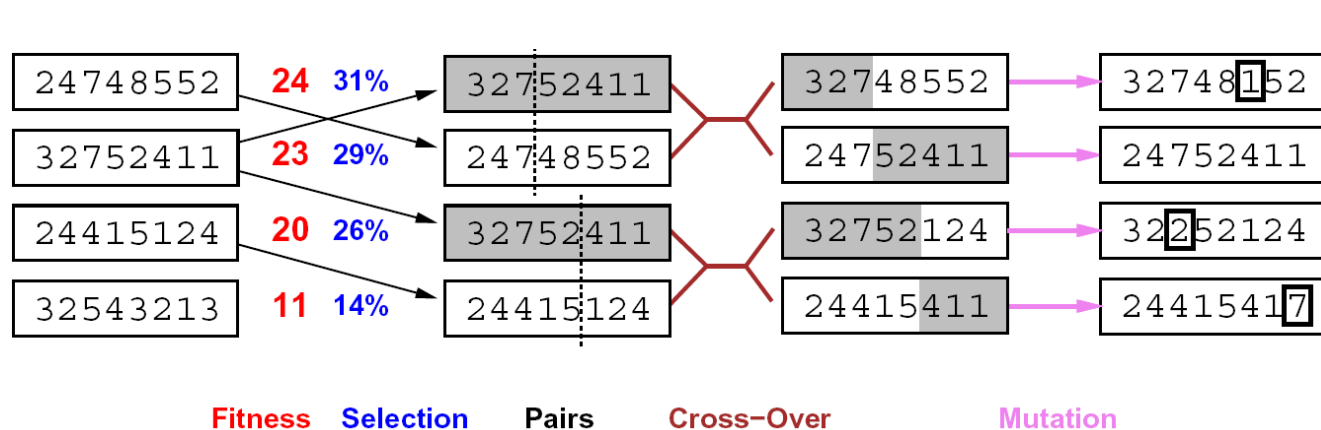
- 搜索间相互 **交流!**

还有什么其他的著名算法采用相同的思想?

- 进化算法!



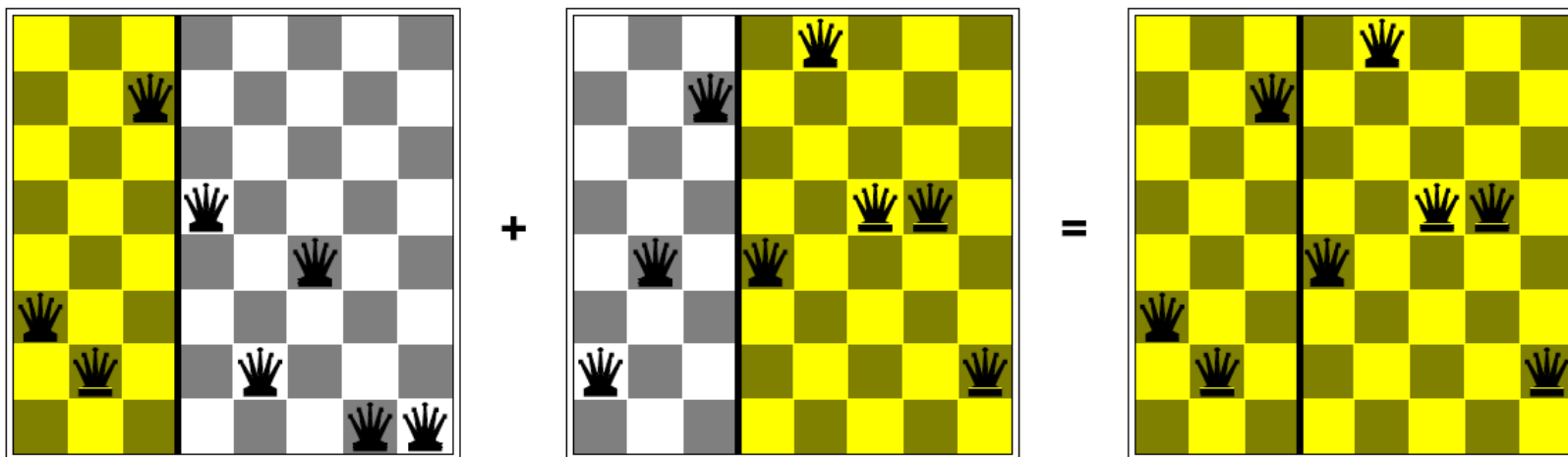
基因遗传算法Genetic algorithms



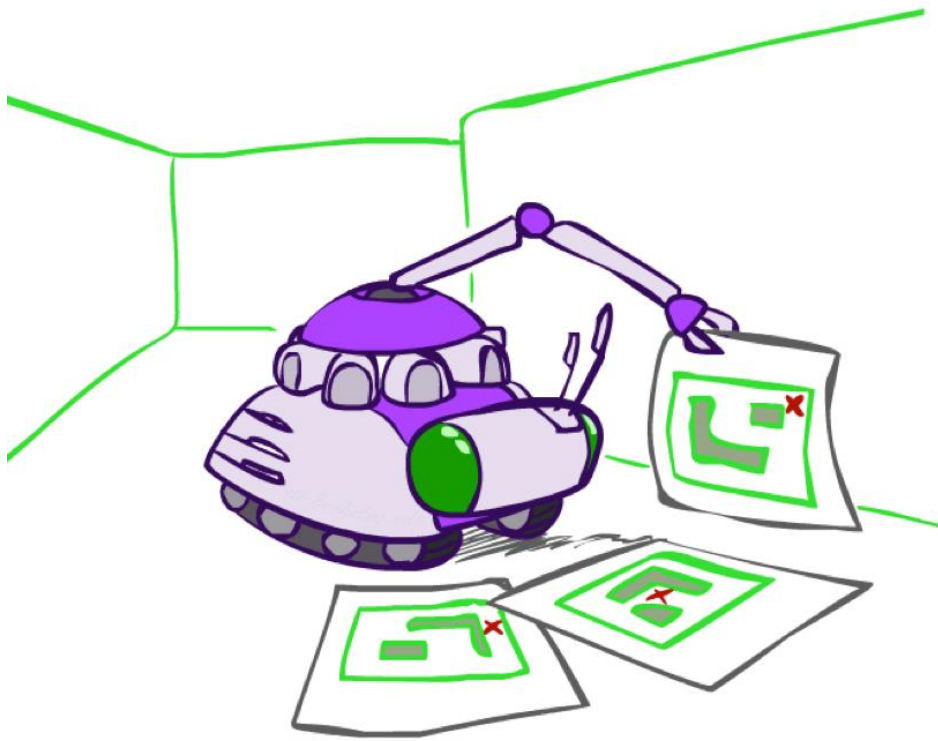
启发于自然选择

- 在每一步选择保持 N 个最好的个体，基于目标函数a fitness function
- 个体对的操作：交错，随机翻转，产生变化的个体

举例: 8皇后问题



不确定搜索



在真实世界里搜索

非确定性: 行动有不可预测的影响

- 问题构建需允许多个行动输出
- 解则是 **条件化的规划**
- 新算法: 与或搜索 (AND-OR)
- 解规划中也可能有循环步骤!

部分可观察性: 感知的不是整个状态

- 可信状态 (belief state) = 智能体可能处在的状态集合
- 问题需在可信状态空间搜索

非确定性 **和** 部分可观察性

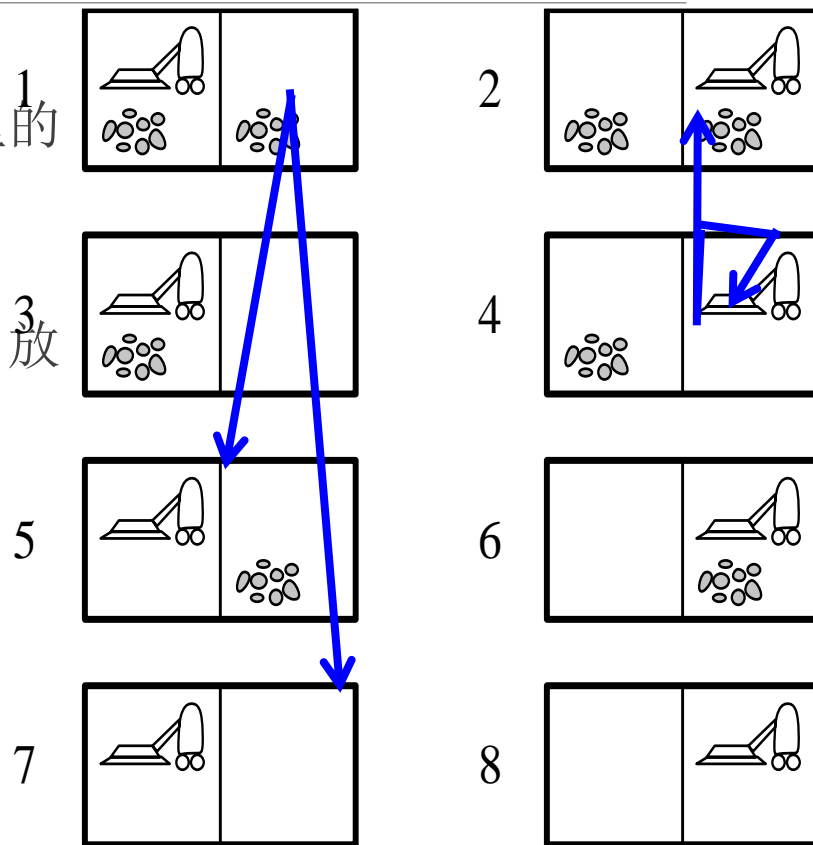
会出故障的吸尘器（行动的结果不确定性造成的）

如果格子有灰尘，**吸尘**可能也吸掉邻近格子里的¹灰尘

- 例如，状态 1 可能会到 5 或 7

如果格子是干净的，**吸尘**也可能发生故障，³放些灰尘在上面

- 例如，状态 4 可能会到 4 或 2

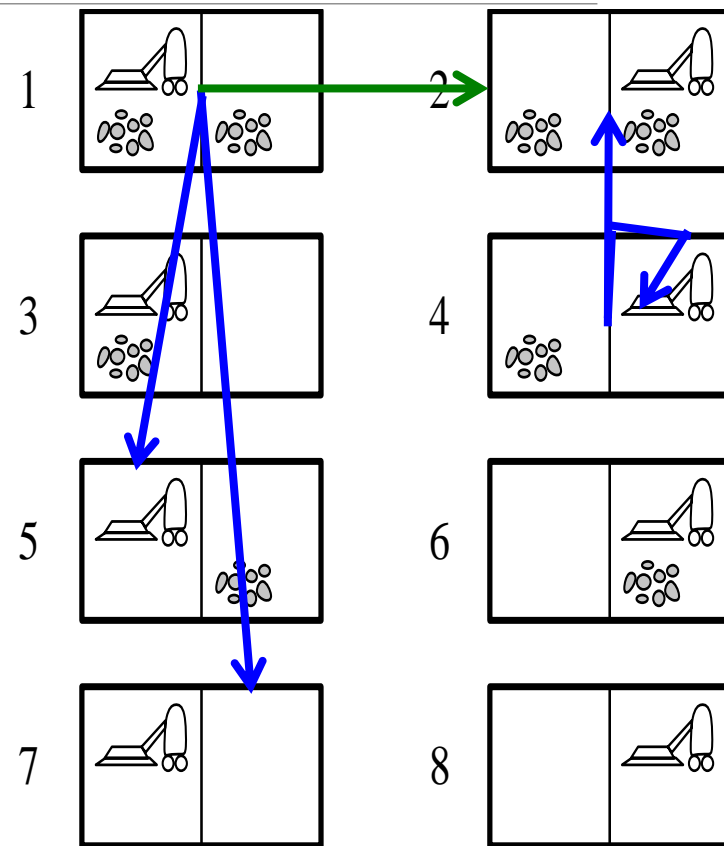


问题建立 (problem formulation)

$Results(s,a)$ 返回一个状态 **集合**

- $Results(1, \text{吸尘}) = \{5, 7\}$
- $Results(4, \text{吸尘}) = \{2, 4\}$
- $Results(1, \text{向右}) = \{2\}$

其他的都和以前一样



条件化的解

从状态 1, 行动 [吸尘] 能解决问题吗?

- 不必然!

那么 [吸尘, 右移, 吸尘]?

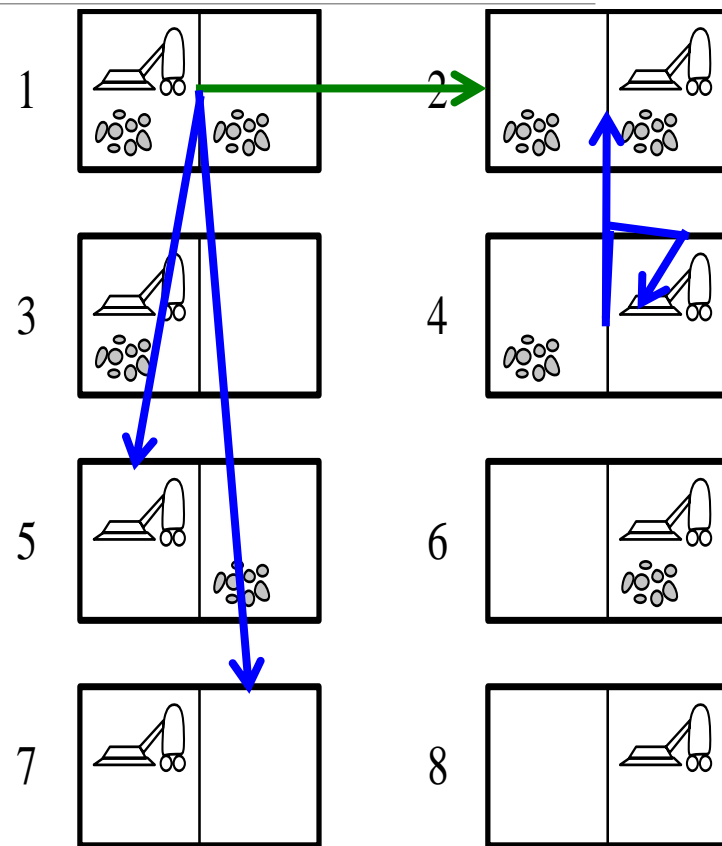
- 不必然!

[吸尘; **if** 状态=5 **then** [右移, 吸尘] **else** []]

这是一个 **依情况而定的解** (分支化或 **条件性规划**)

如何找到这样的解?

这是一个解



与或搜索树(And-Or search trees)

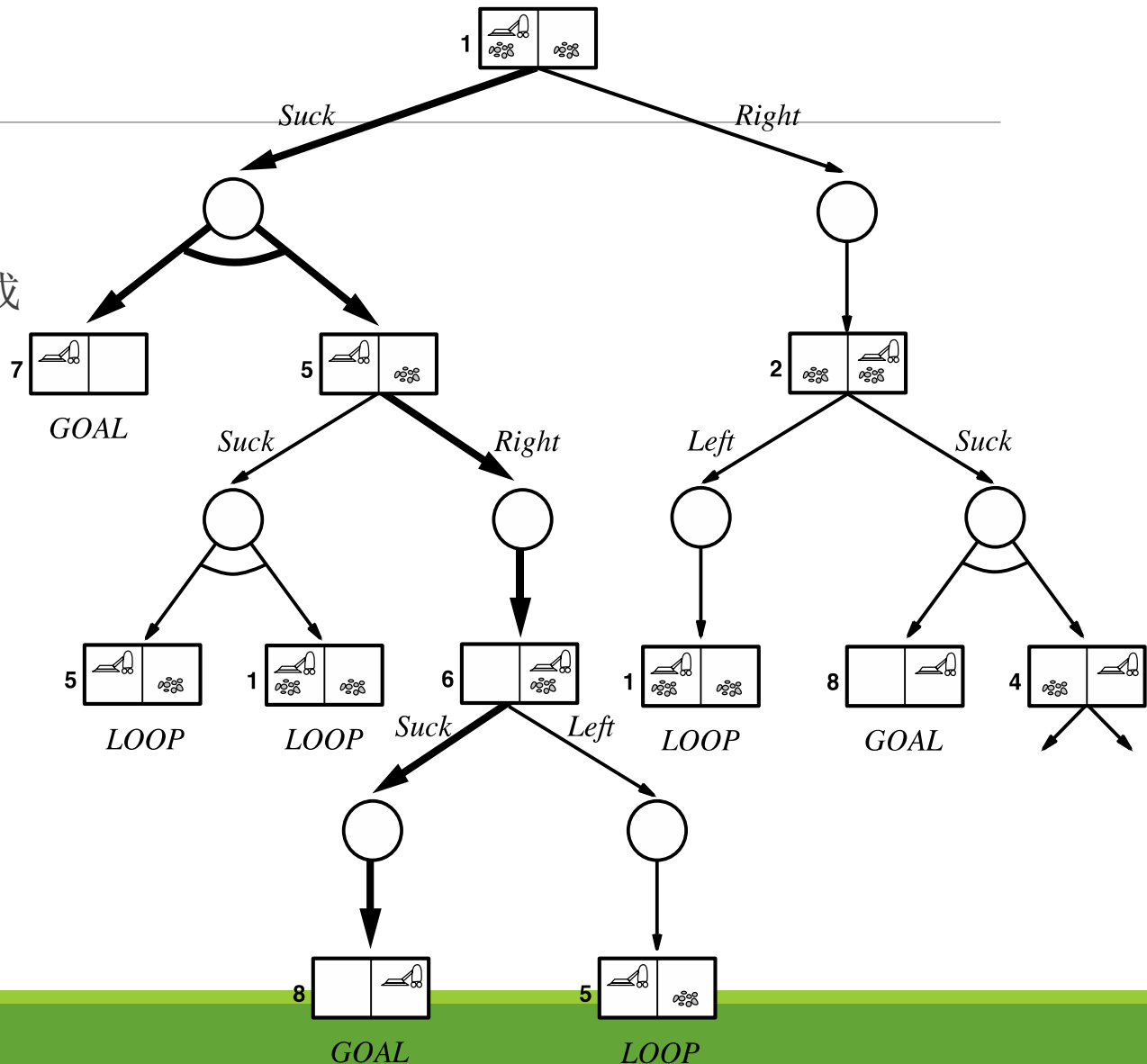
或-节点:

- 智能体选择行动;
- 至少一个分支**能得到解即成功

与-节点:

- 自然选择行动的影响;
- 所有分支**必须都能被求解, 否则失败

两种节点按层交替排列



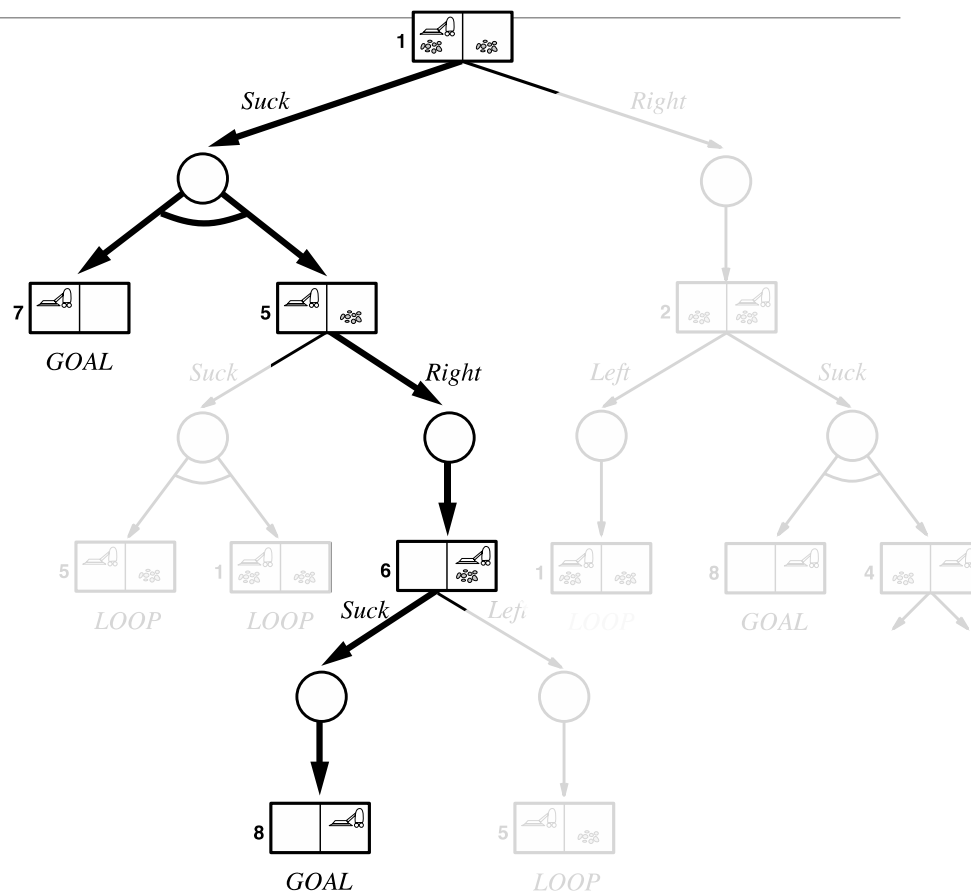
与或搜索树

树节点

- 选择不同的行动 (OR)
- 同一行动可能的不同结果 (AND)

条件化的搜索解

- 还是一个树，条件化选择行动 (if this, do that, else if...)
- 也有可能存在循环



AND-OR 搜索策略

AND-OR search: 调用 OR-Search 在根节点上

OR-search(node): 如果在任意一个行动的结果集合上 AND-search 获得成功，则返回成功

AND-search(set of nodes): 如果OR-search 在这个集合内的所有节点上都成功，则返回成功

与或搜索算法（递归，深度优先）

Function 与或图搜索(问题) **returns** 一个条件性规划, 或失败

或-搜索(问题.初始状态, 问题, [])

Function 或-搜索(状态, 问题, 路径) **returns** 一个条件性规划, 或失败

if 问题.goal-test(状态) **then return** 空规划

if 状态 曾出现在 路径 **then return** 失败

for each 行动 **in** 问题.actions(状态) **do**

规划 \leftarrow 与-搜索(results(状态,行动),问题,[状态 | 路径])

if 规划 \neq 失败 **then return** [行动 | 规划]

return 失败

一个行动上的与搜索成功，则或搜索返回包含该行动的一个规划

与或搜索，继续

Function 与-搜索(状态集, 问题, 状态路径) **returns** 一个条件规划, 或失败

for each s_i **in** 状态集 **do**

 行动 $\text{plan}_i \leftarrow \text{OR-SEARCH}(s_i, \text{问题}, \text{路径})$

if $\text{plan}_i = \text{失败}$ **then return** 失败

return [**if** s_1 **then** plan_1

else if s_2 **then** plan_2

else ... **if** s_{n-1} **then** plan_{n-1} **else** plan_n]

如果有一个结果状态节点的或搜索返回失败，则其父节点的与搜索返回失败，说明该行动方向不成功。

会打滑的吸尘器

有时移动可能会不成功（停在原地）

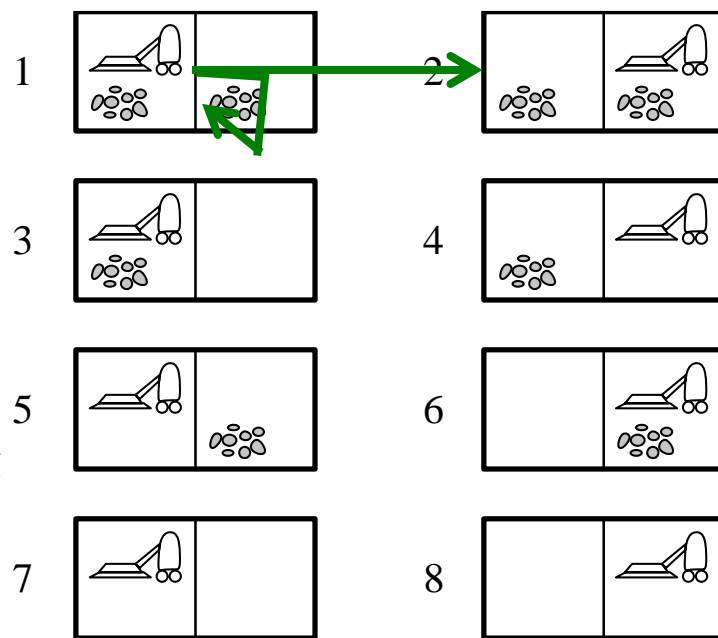
不能保证会得到一个条件化的解！

存在一个 **循环解 cyclic solution**:

- [Suck, L1]
- 这里 L1 只是一个标签，它代表：
 - L1: Right, if State = 5 then L1 else Suck
- 修改 AND-OR-GRAPH-SEARCH，当找到一个重复状态时，5 添加一个标签，增加一个条件分支指向这个标签，而不是返回一个失败。（使得相应动作可重复执行）

一个循环行动规划是一个循环解的条件是：

- 每个叶节点是一个目标状态
- 在该循环规划里的任一点，都有一条路径走到一个叶节点



非决定性 nondeterminism 到底表示的是什么？

举例：你的旅店门卡不能打开你旅店房间的门

- 解释 1: 你没有用对方式
 - 这是 *nondeterminism*; 不断的尝试
- 解释 2: 门卡本身有问题
 - 这是 *partial observability*; 需要换一个新的门卡
- 解释 3: 这不是你的房间
 - 这是 *embarrassing*; *be ashamed*

当输出结果不决定性的依赖于某个隐藏状态，
这个非决定性的模型才是合适的



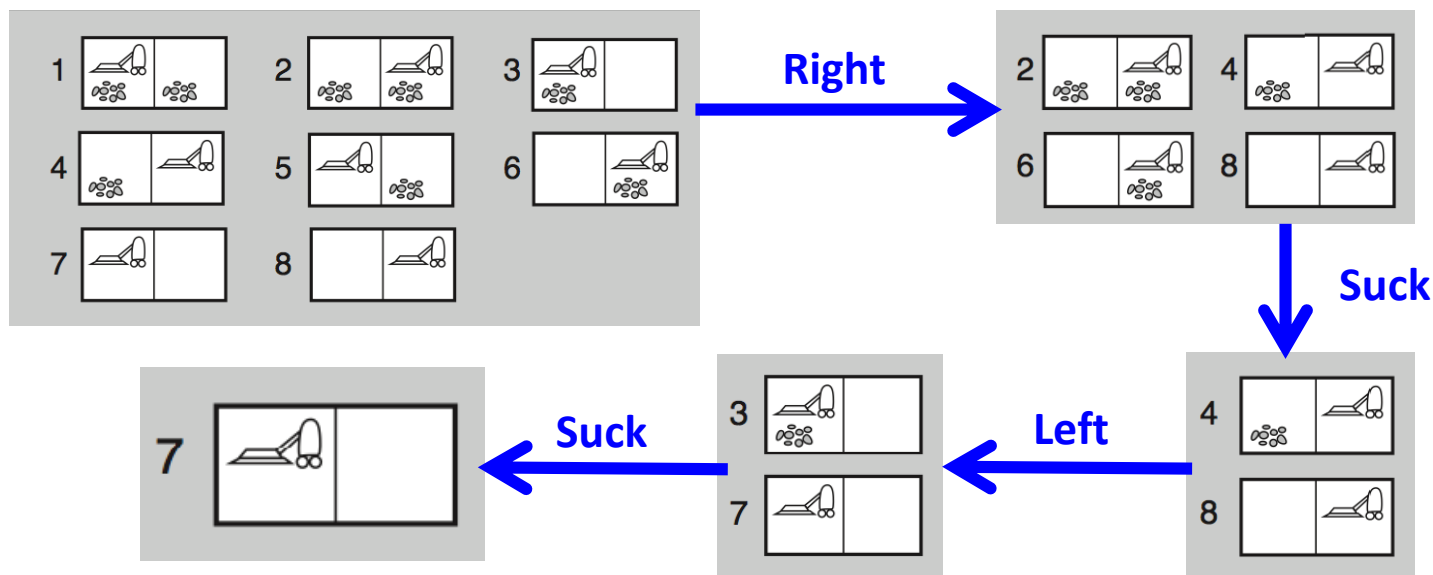
部分可观察性 Partial observability

极端的 部分可观察性: 没传感器的环境世界

在吸尘器例子里，假设没有传感器！

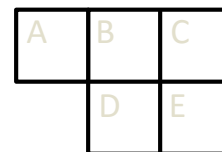
可信状态 *belief state*: 一个集合包括智能体可能处在的所有环境状态

- 通常情况, 给定当下的感知，智能体所知道的情况



状态, 可信状态, 可信状态空间

例如: 5 个网格位置 {A,B,C,D,E} 可能有 0-5 个妖怪



状态空间的表达:

- 5 布尔值 (a,b,c,d,e)

状态空间大小:

- 2^5

状态举例:

(1,1,1,1,1) ghosts everywhere
(0,0,0,0,0) no ghosts
(1,1,0,0,0) ghosts in just A and B

可信状态举例:

I believe that there are ghosts
everywhere or no ghosts at all

(1,1,1,1,1)

(0,0,0,0,0)

Empty belief state

I believe that there is exactly
one ghost

(1,0,0,0,0) (0,1,0,0,0)

(0,0,1,0,0) (0,0,0,0,1)

(0,0,0,0,1)

Every configuration is possible

(0,0,0,0,0) (1,0,0,0,0) ...

(0,1,0,0,0) (1,1,0,0,0) ...

(0,0,1,0,0) (1,0,1,0,0) ...

(0,1,1,0,0) (1,1,1,0,0) ...

(0,0,0,1,0) (1,0,0,1,0) ...

...

...

无传感器问题的构建

底层的“物理”问题通常有 Actions_p , Result_p , Goal-Test_p , 和 Step-Cost_p .

初始状态: 一个可信状态(belief state) b (包括若干或所有物理状态 s)

- N 个物理状态(physical states) $\Rightarrow 2^N$ 可信状态 (belief states)

Goal test: 每个元素状态 s in b 满足 $\text{Goal-Test}_p(s)$

Actions: 行动合集 $\text{Actions}_p(s)$ for each s in b

转移模型 Transition model:

- 决定性的 Deterministic: $\text{Result}(b,a) = \text{结果状态集合 } \text{Result}_p(s,a) \text{ for each } s \text{ in } b$
- 非决定性的 Nondeterministic: $\text{Result}(b,a) = \text{结果状态集合 } \text{Results}_p(s,a) \text{ for each } s \text{ in } b$
- $\text{Step-Cost}(b,a,b') = \text{Step-Cost}_p(s,a,s')$ for any s in b
- 假设给定行动 a , 步骤成本对于任何 s in b 都相同

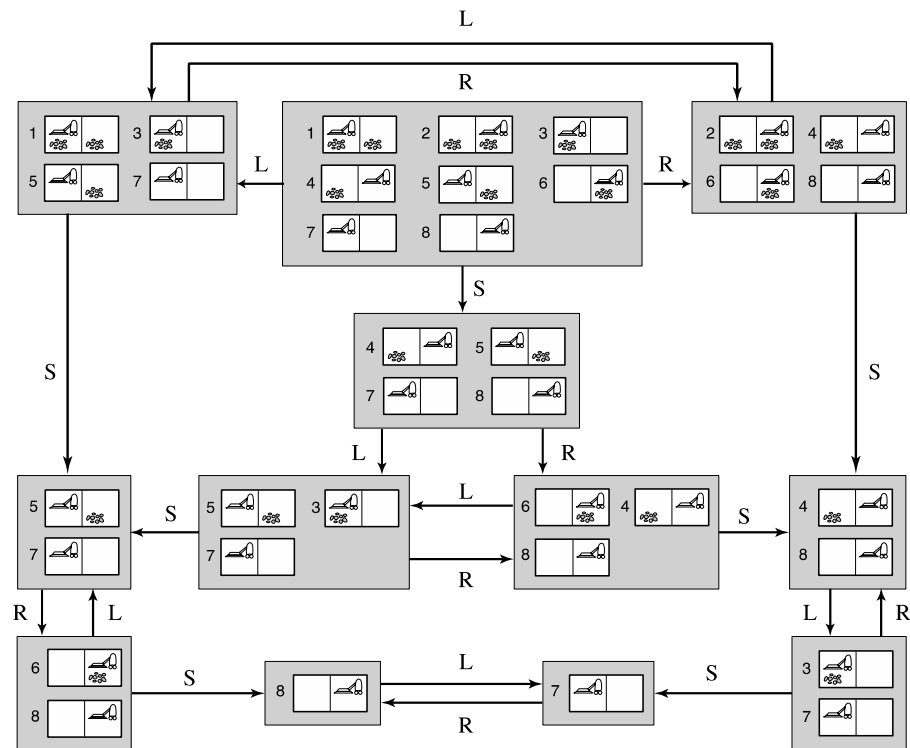
在无传感器可信状态空间里的搜索

所有搜索过程都和以前一样!

解还是行动的序列!

一些变通的地方:

- If any s in b is unsolvable, b is unsolvable
- If b' 包含 b and b' 有一个解, b 有同样的解
- If b' 包含 b , 并且在搜索树里我们已找到一条路径通向 b , 那么可以丢弃 b'



书上图4.14, 决定性行动的, 无感知, 可信状态空间的状态图

无传感器问题有什么现实意义?

反映了许多现实中机器人的操控问题

“零件导向传送器”由一系列倾斜的导向装置组成，可以把零件按一个方向进行导向，无论它们的初始方向是什么

这比起应用摄像机和机器人手臂更加简单方便，且更可靠!



播放演示视频

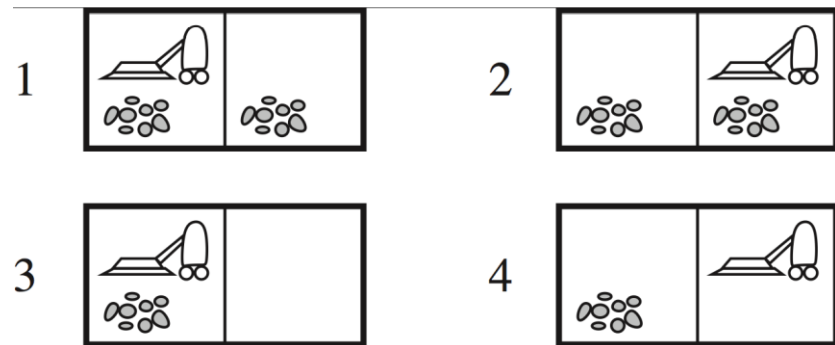
部分可观察性Partial observability: 问题建立

需说明智能体能观测到什么:

- 行动决定性的 **Deterministic**: $\text{Percept}(s)$ 是在状态 s 收到的感知
- 非决定性的 **Nondeterministic**: $\text{Percepts}(s)$ 在状态 s 收到的可能的感知
- 全部可感知: $\text{Percept}(s) = s$ 无传感器: $\text{Percept}(s) = \text{null}$

■ 智能感知局部（当前房间的）吸尘器环境世界

- $\text{Percept}(s_1) = [A, \text{Dirty}]$
- $\text{Percept}(s_3) = [A, \text{Dirty}]$



部分可观察性: 可信状态转移模型 belief state transition model

$b' = \text{Predict}(b, a)$ 更新可信状态 *只根据行动 a*

$\text{Possible-Percepts}(b')$ 是在更新后的可信状态下可能导致的感知的集合

- Union of $\text{Percept}(s)$ for every s in b'

$\text{Update}(b', p)$ 是新的可信状态, 如果收到的是感知 p (根据感知对可信状态过滤)

- 仅包含状态 s in b' for which $p = \text{Percept}(s)$

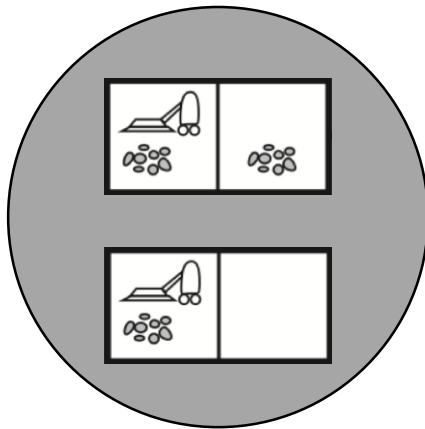
$\text{Results}(b, a)$ 返回结果的可信状态集合

- $\text{Update}(\text{Predict}(b, a), p)$ for each p in $\text{Possible-Percepts}(\text{Predict}(b, a))$

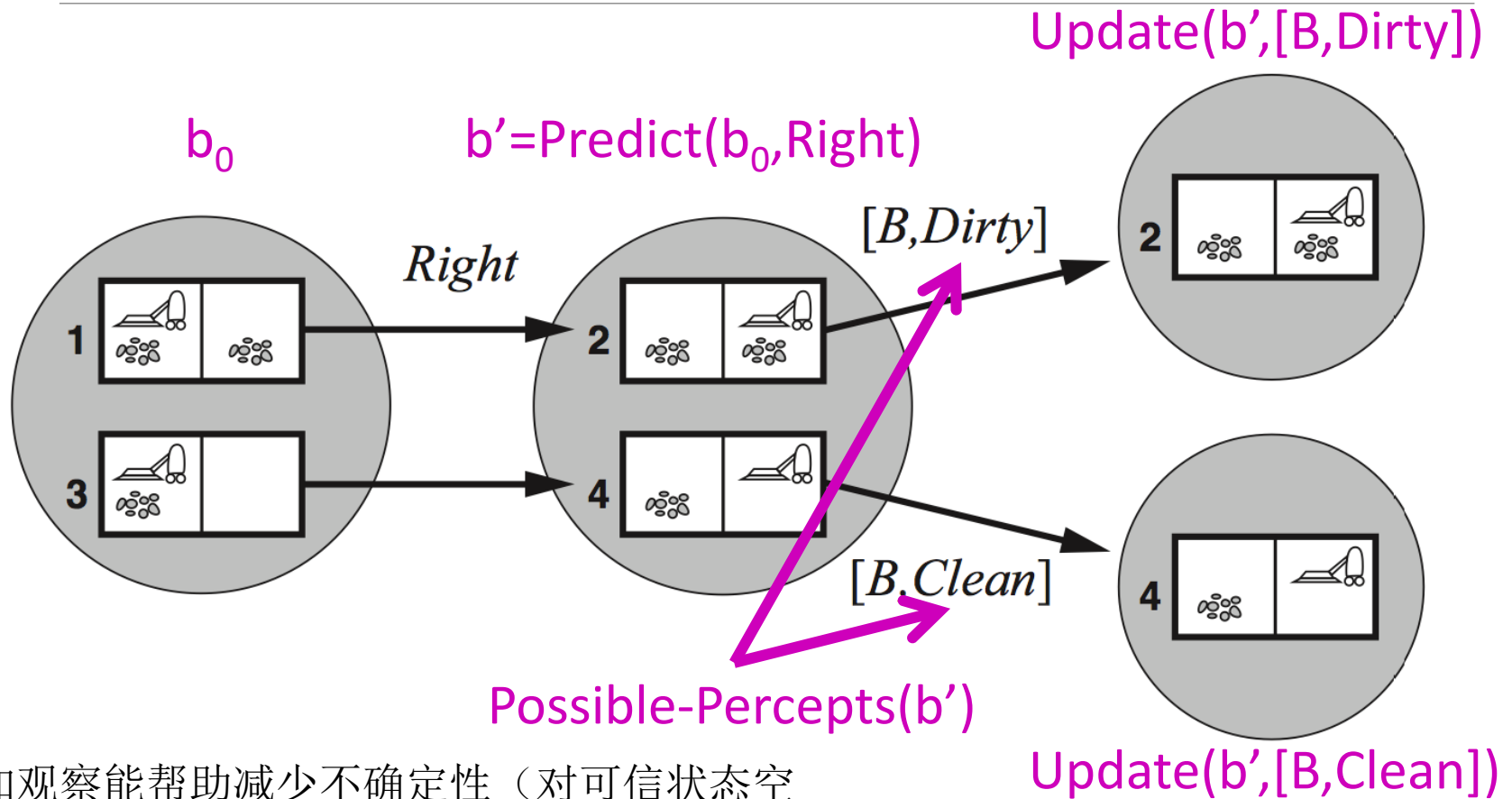
举例: Results(b_0 , Right)

b_0

Right



举例: $\text{Results}(b_0, \text{Right})$



感知观察能帮助减少不确定性（对可信状态空间进行过滤）

智能体对可信状态的修改 (维护)

感知 p 取决于环境

递归的可信状态估计（计算根据上一个可信状态）：

- $b \leftarrow \text{Update}(\text{Predict}(b,a),p)$

形成了 预测-更新循环 ***predict-update cycle***

- 类似的称谓： ***monitoring, filtering, state estimation***
- 定位 ***Localization*** 和 画地图 ***mapping*** 是两种特殊的应用

总结

非确定性（行动导致的） 要求解是条件化的规划
通过与或搜索寻找解

无感知问题 的解是通常的行动规划
通过在信念状态空间中去寻找解

通常的部分可观察环境 导致 感知上的非确定性
在信念状态空间里进行与或搜索（And-Or search）

通过 预测-更新 循环 (Predict-Update cycle) 进行可信状态的转移

