从英语翻译成中文(简体) - www.onlinedoctranslator.com

公告

§ HW1 到期**1 月 30 日，星期二，**

太平洋时间晚上 11:59

§ 项目 1 到期**2 月 2 日，星期五，**

太平洋时间晚上 11:59

立即预扫描考勤二维码！

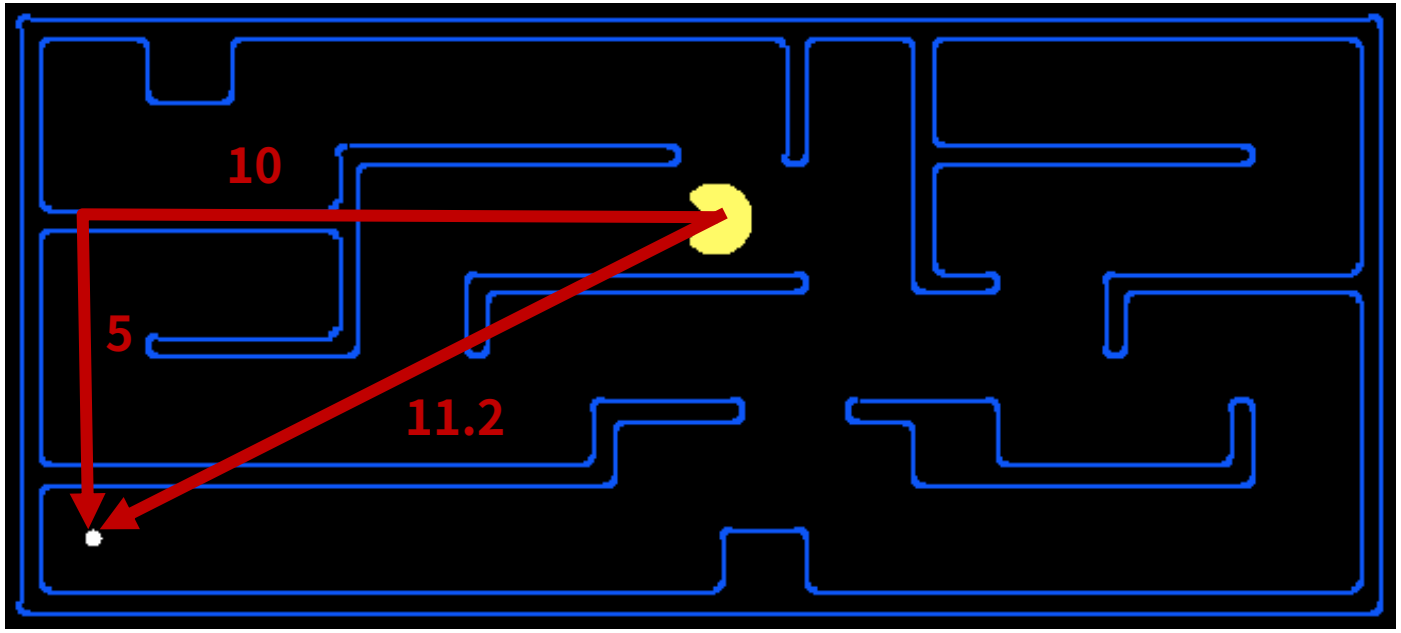
（密码稍后出现）

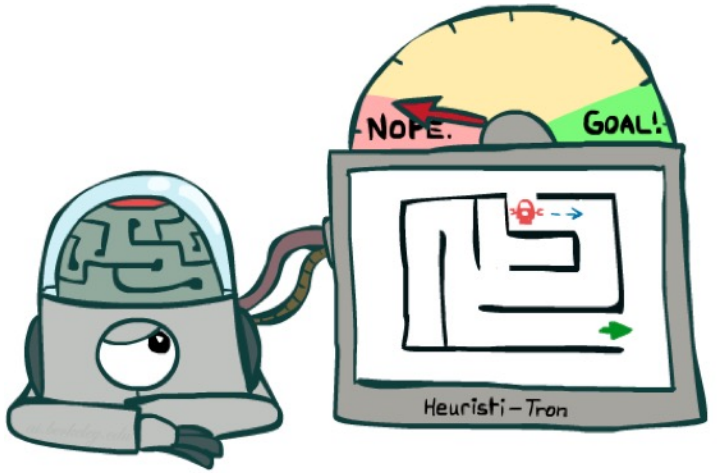
[这些幻灯片由 Dan Klein 和 Pieter Abbeel 为加州大学伯克利分校的 CS188 人工智能入门课程制作。所有 CS188 材料均可在 http://ai.berkeley.edu 获取。]
[更新幻灯片来自：Stuart Russell 和 Dawn Song]

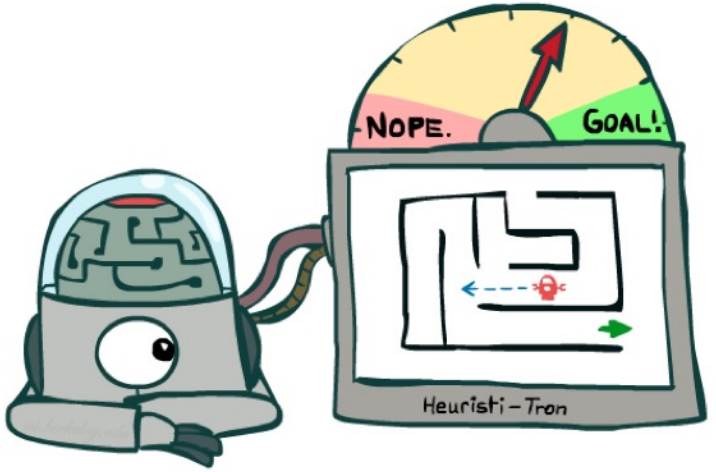
回顾：搜索启发式

§ 启发式方法是：

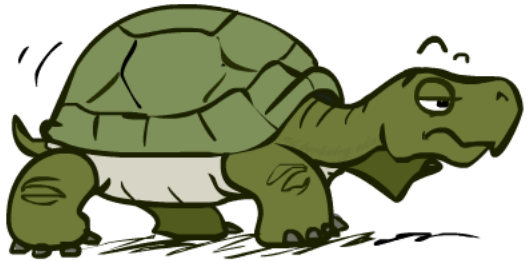
- § 一个函数*估计*某个状态与目标的接近程度 专为特定搜
- § 索问题而设计
- § 示例：曼哈顿距离、路径欧几里得距离





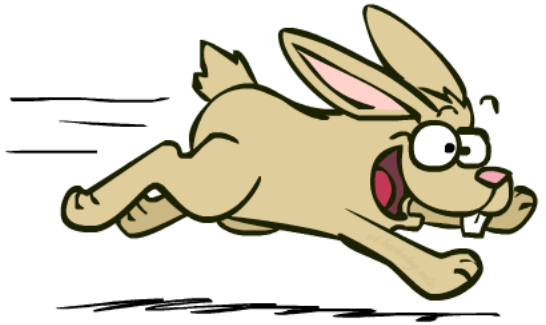


回顾：



统一成本搜索

（仅成本，g）



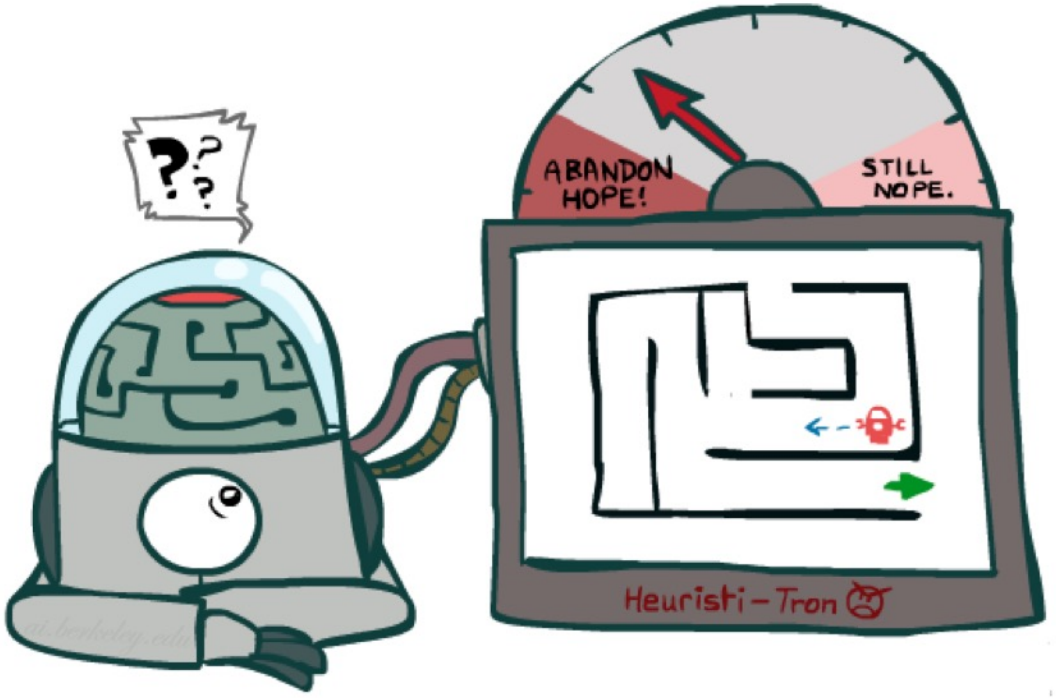
贪婪最佳优先搜索

（仅启发式，h）

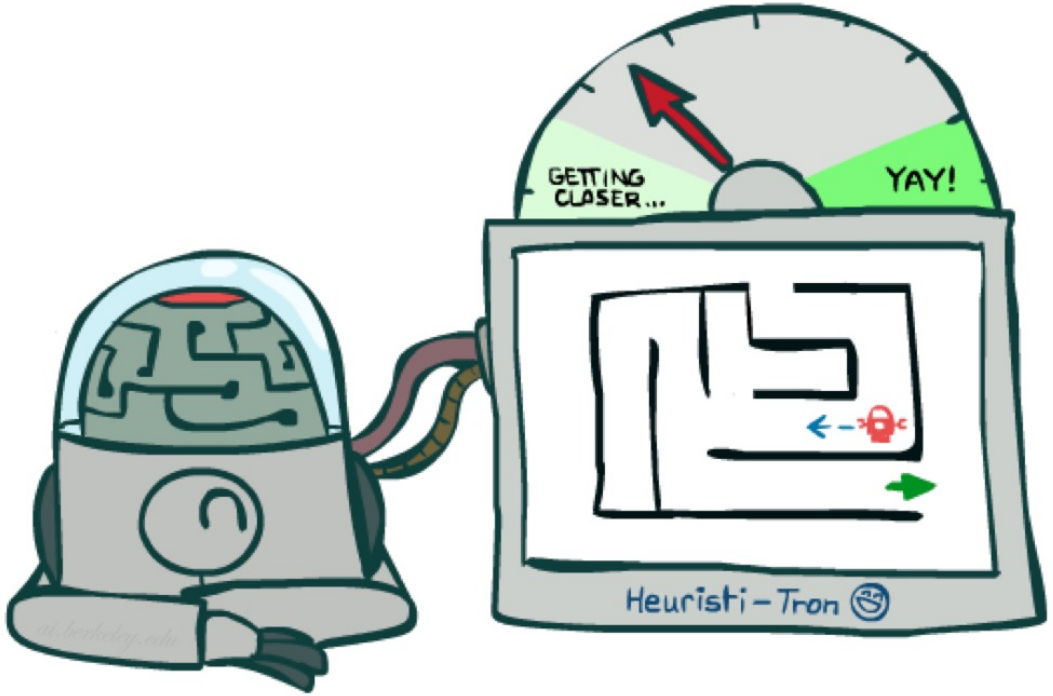
A* 搜索

（两者，f=g+h）

回顾：可接受性

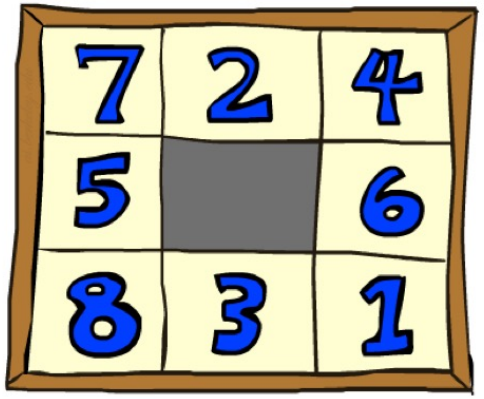


不可接受的（悲观的）启发式方法会将好的计划困在边缘，从而破坏最优性

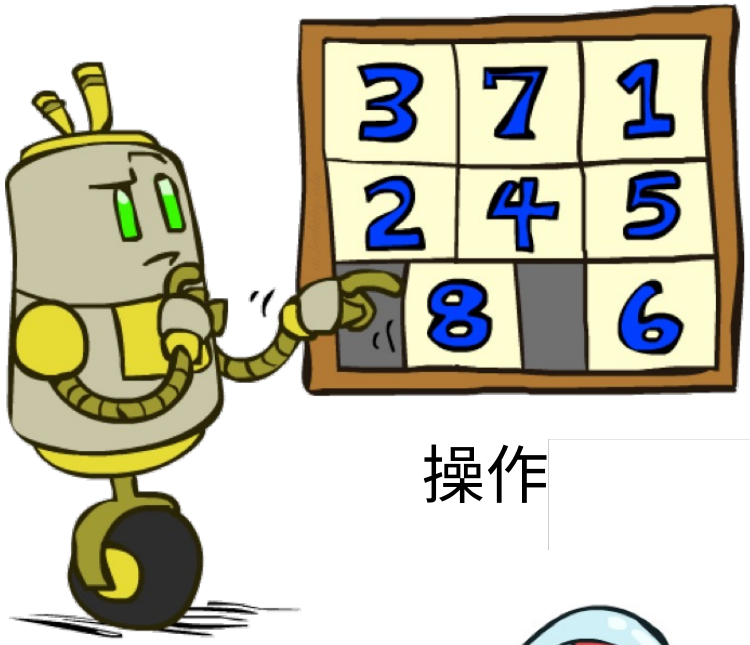


可接受的（乐观的）启发式方法减慢糟糕的计划，但永远不会超过真正的成本

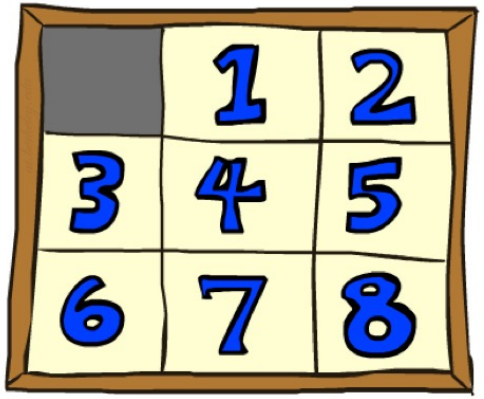
回顾：8 道谜题



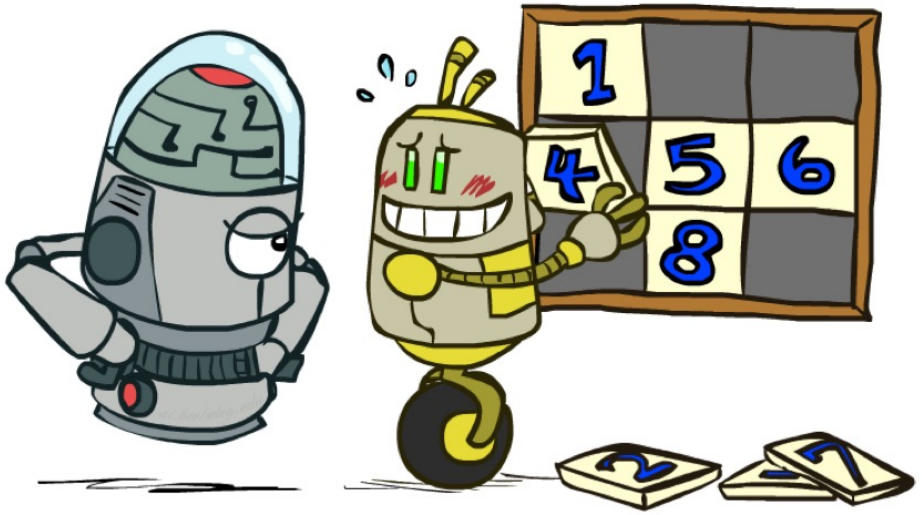
起始状态



操作



目标状态

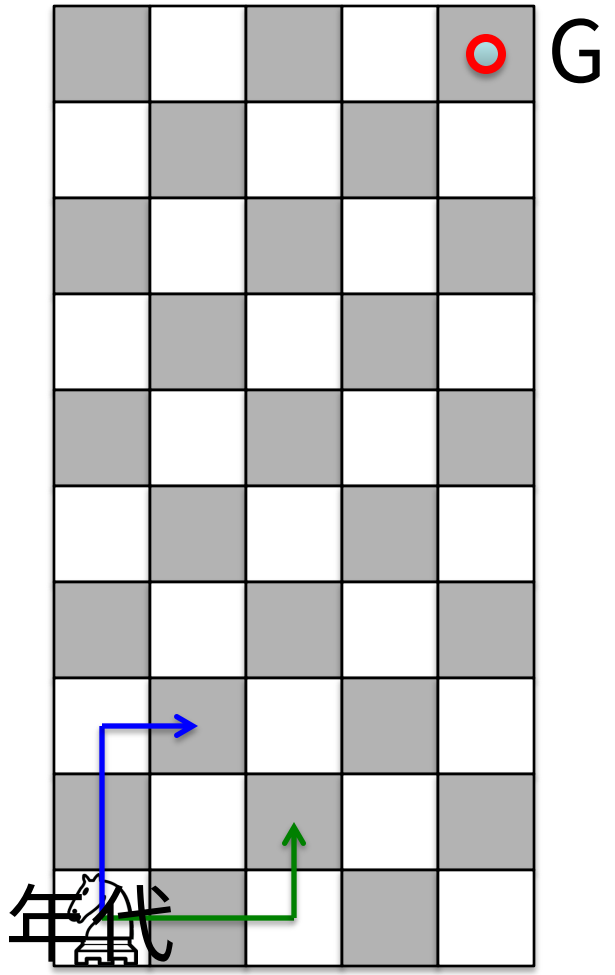


设计启发式方法：骑士的走法

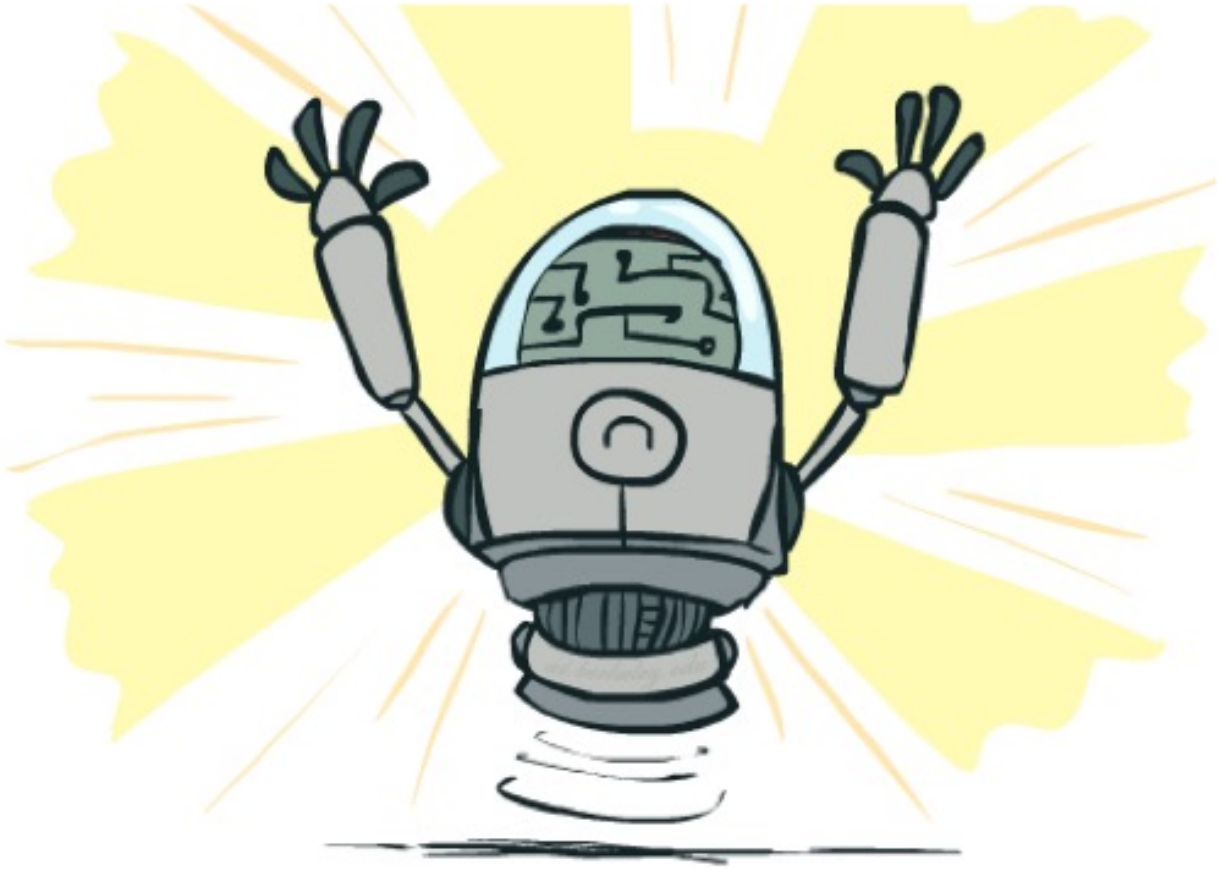
§ 从 S 到 G 至少需要多少骑士移动一次?

- § H_1= (曼哈顿距离)/3
  - § H_1=H_1四舍五入到正确的奇偶校验（即使 S、G 颜色相同，否则为奇数）
- § H_2=（欧几里得距离）/5 $\sqrt{}$
  - § H_2=H_2四舍五入为正确奇偶校验
- § H_3=（最大水平或垂直距离）/2
  - § H_3=H_3四舍五入为正确奇偶校验

§ $H(n)$ = 最大（$H_1'(n), H_2(n), H_3'(n)$）是可以接受的!

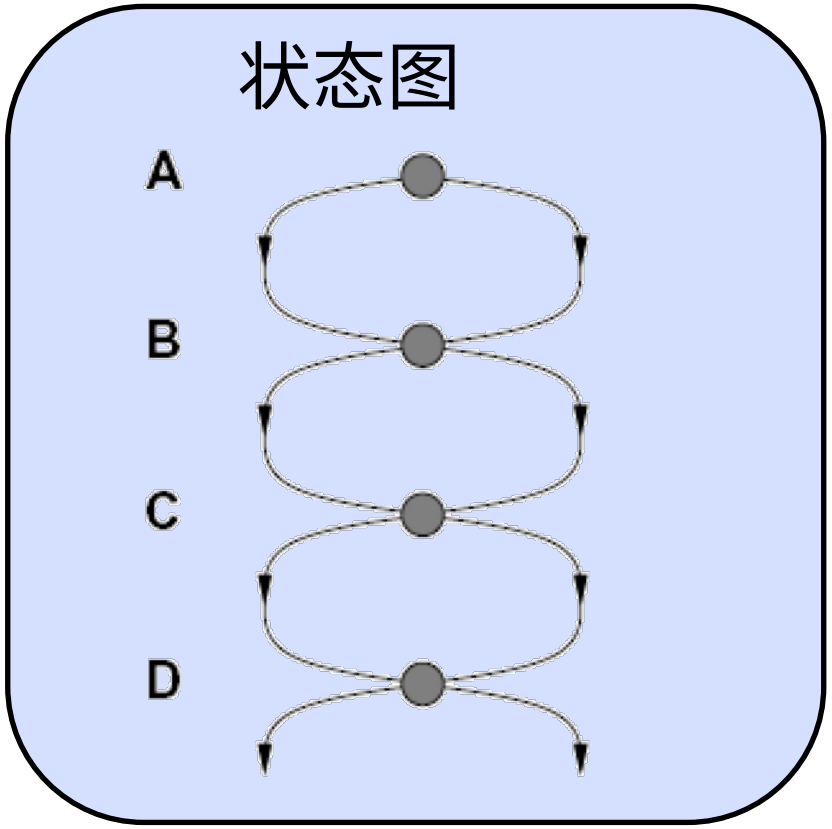


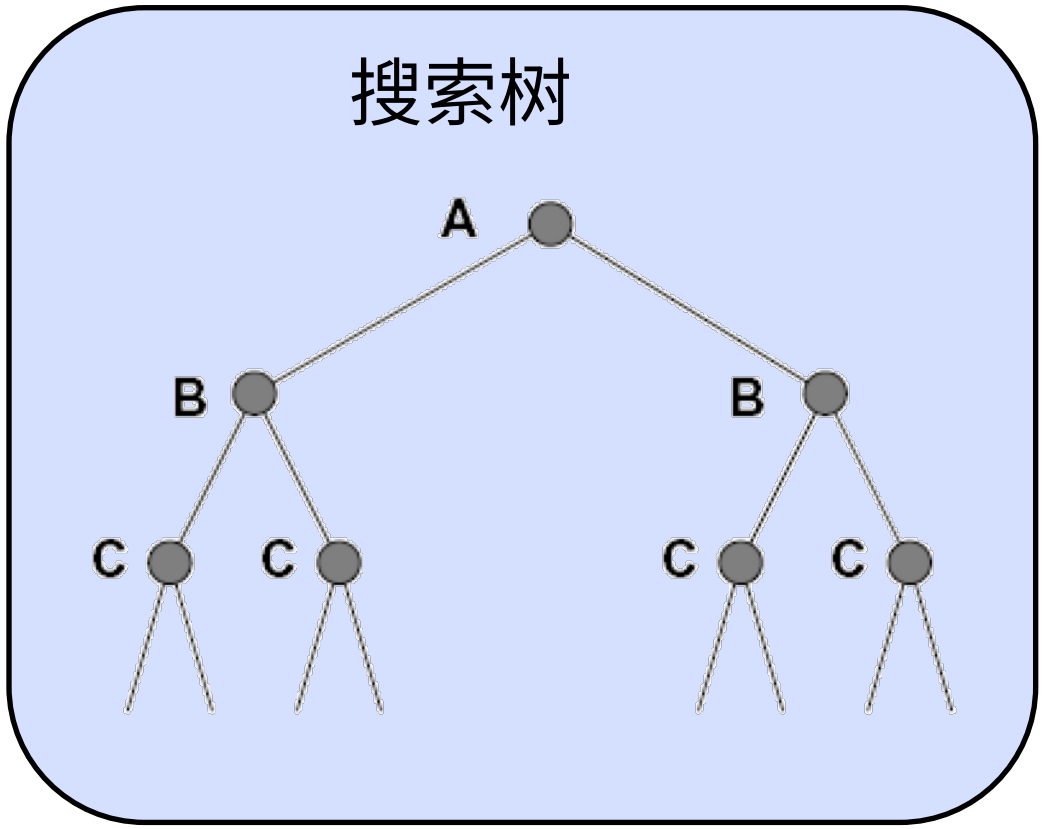
回顾：A* 树搜索的最优性



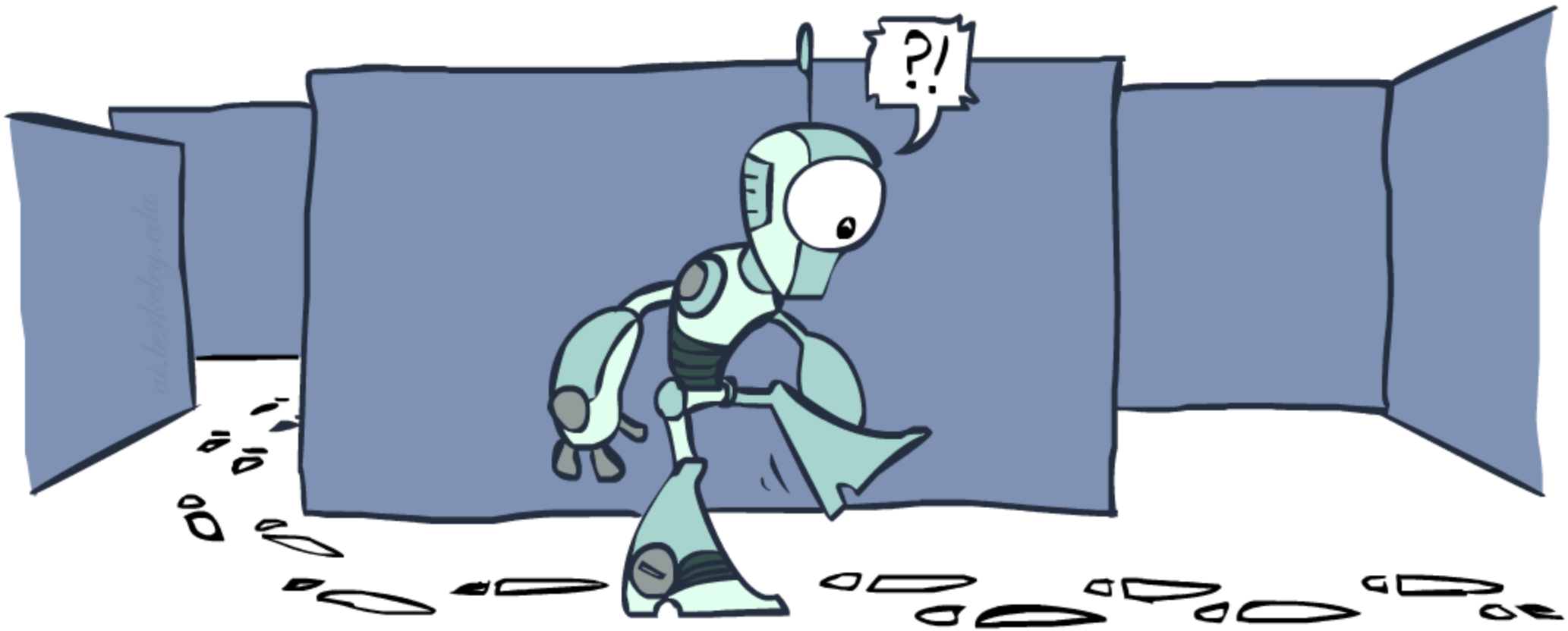
树搜索：额外的工作！

§ 无法检测到重复状态可能会导致工作量成倍增加。



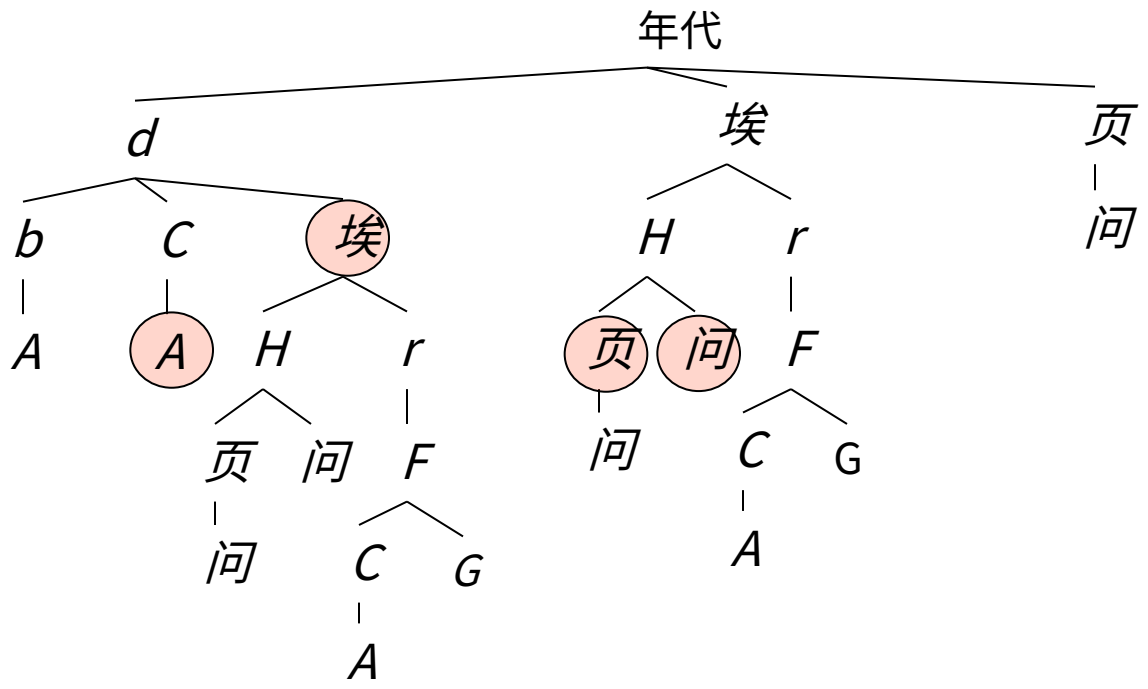


图表搜索



图表搜索

§ 例如，在 BFS 中，我们不应该费心扩展圆圈节点（为什么？）



图表搜索

§ 想法：从不扩张一个州两次

§ 如何实现：

- § 树搜索 + 扩展状态集（“封闭集”）逐个节点扩
- § 展搜索树，但是……
- § 在扩展节点之前，请检查以确保其状态从未被扩展过

- § 如果不是新的，则跳过，如果是新的则添加到封闭集

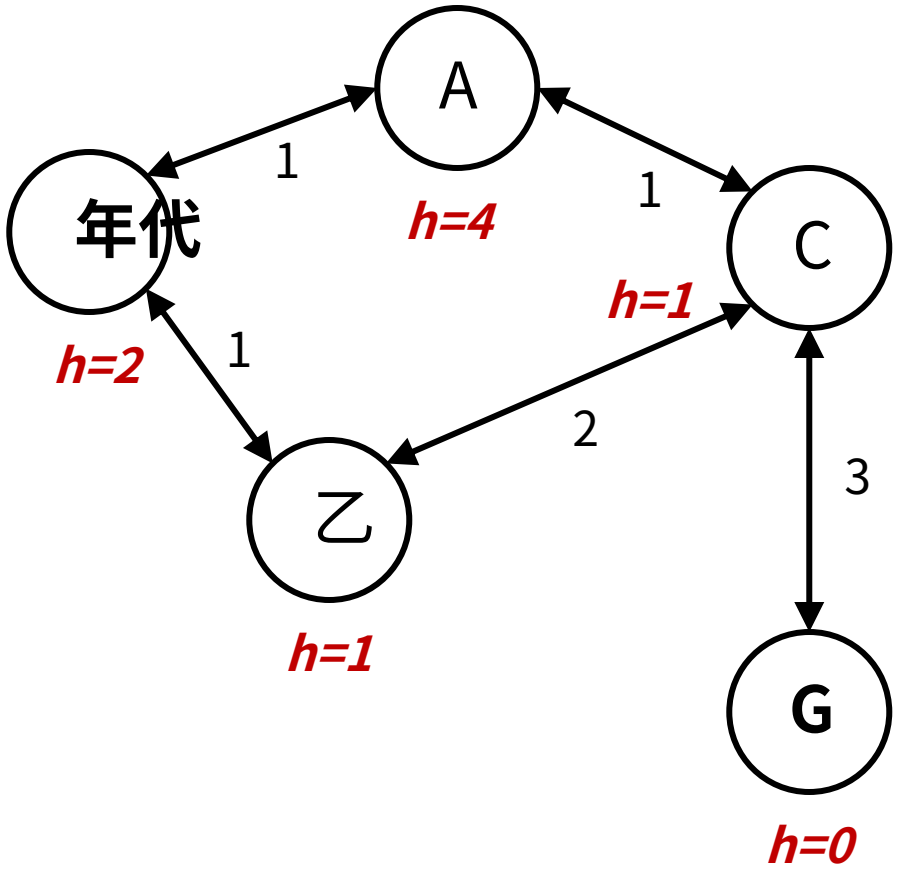
§ 重要的：将闭集存储为一个集合，不是列表

§ 图形搜索会破坏完整性吗？为什么/为什么不？

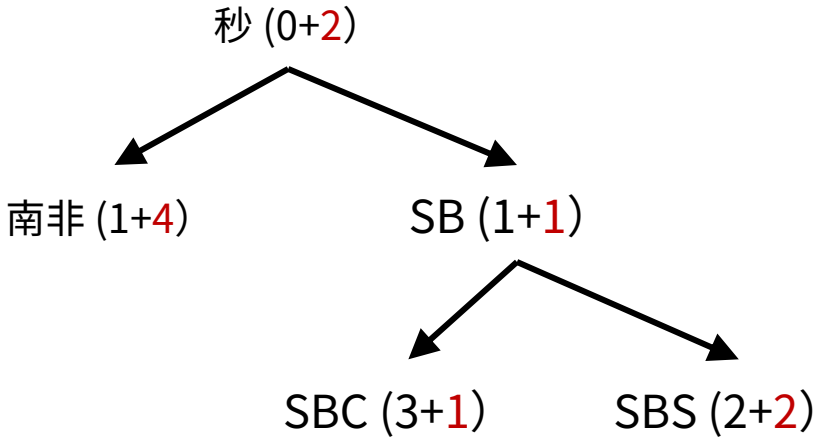
§ 最优性如何？

A* 图搜索出错了吗?

状态空间图



搜索树

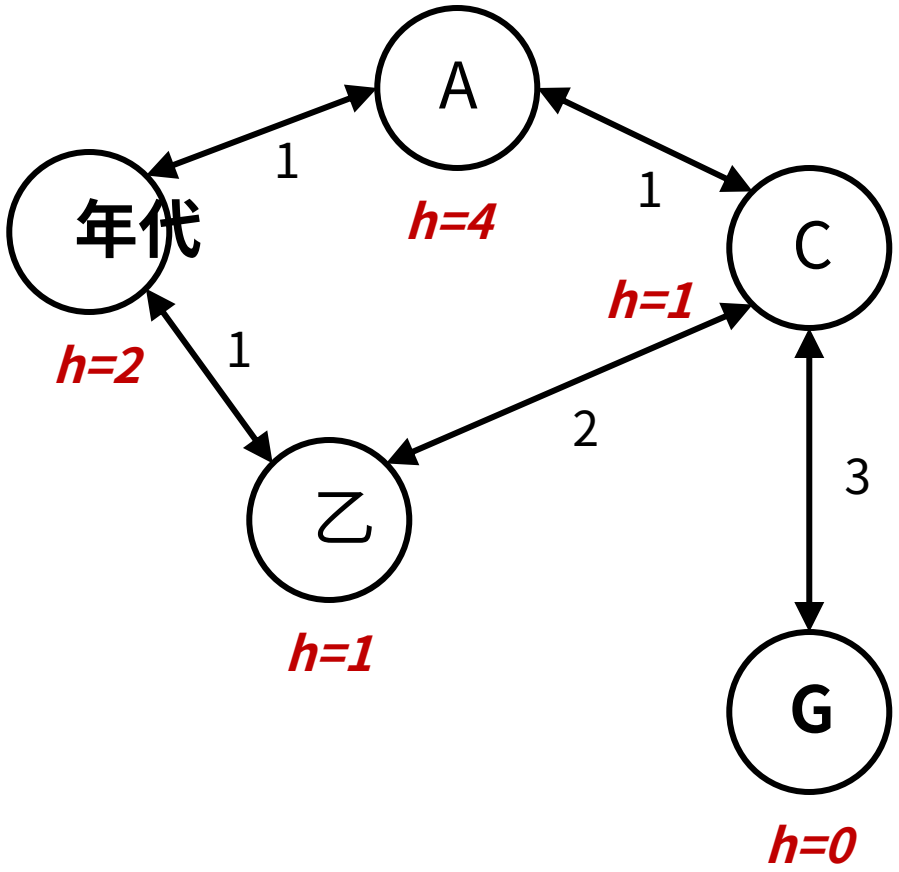


闭集

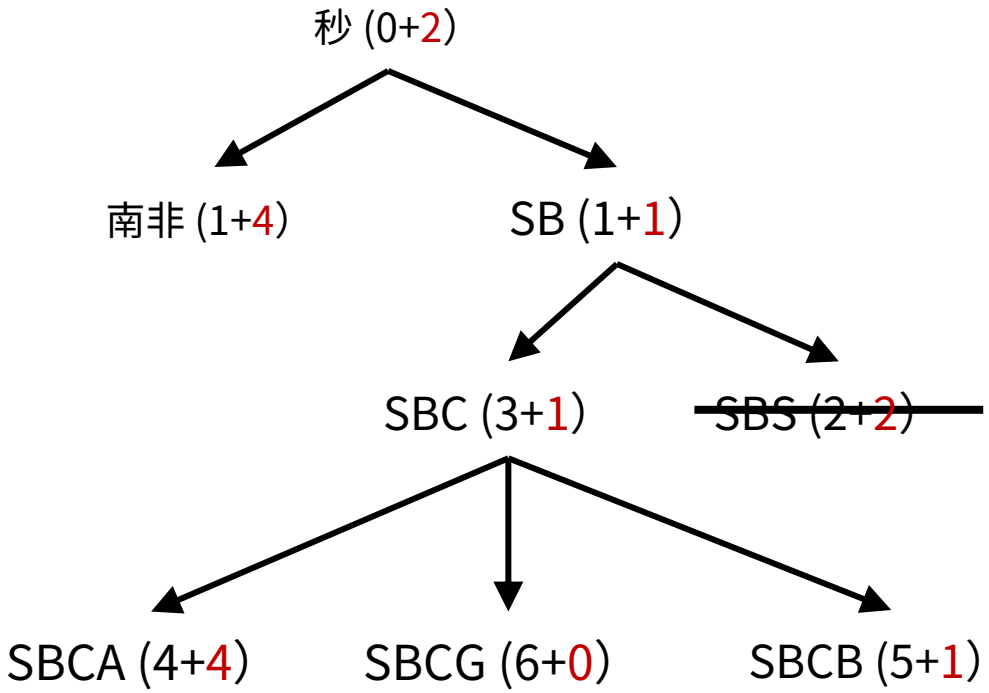
{ 某人 }

A* 图搜索出错了吗?

状态空间图



搜索树

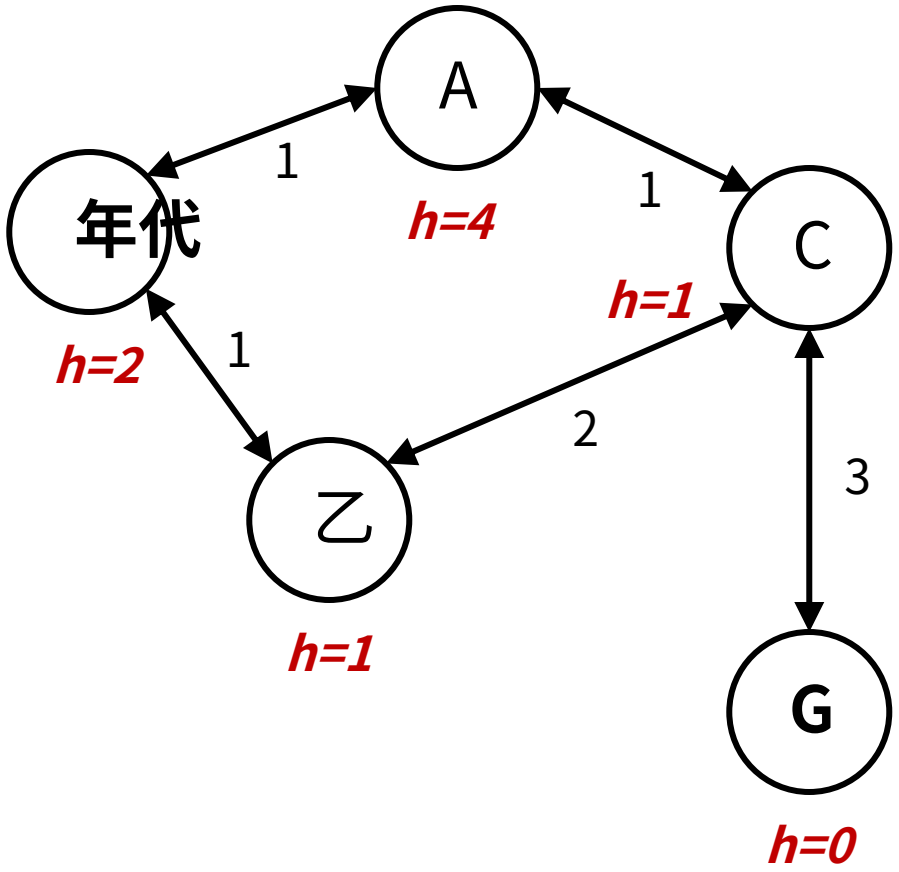


闭集

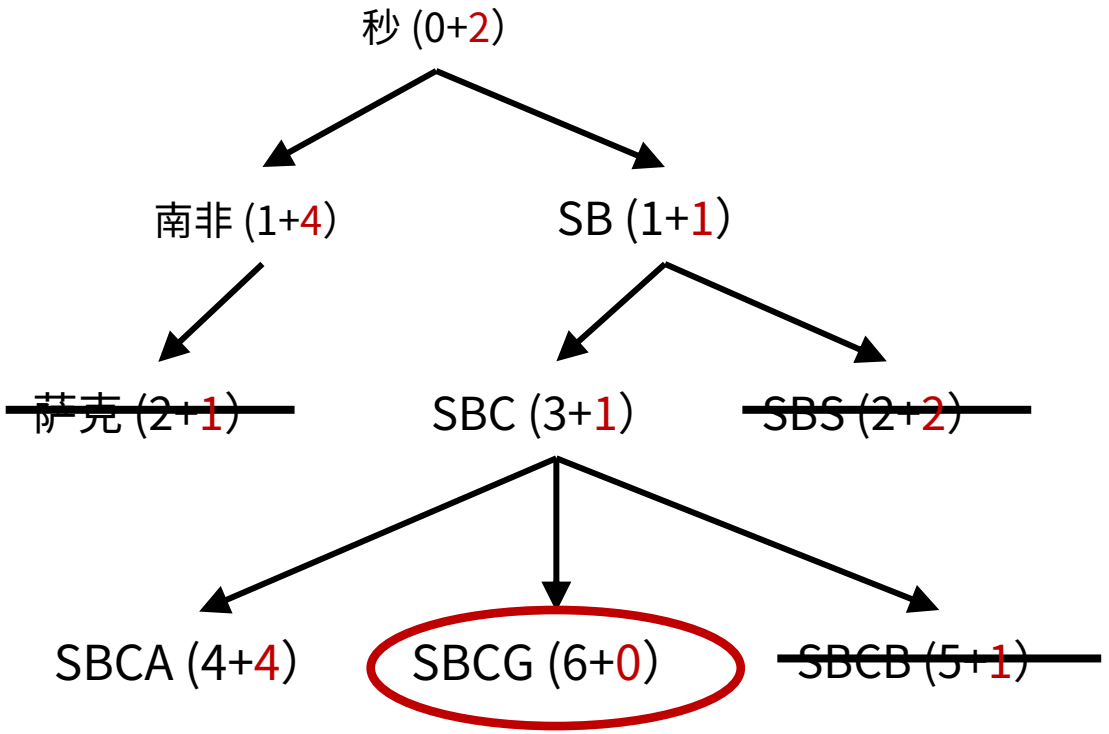
{ 某人 }

A* 图搜索出错了吗?

状态空间图



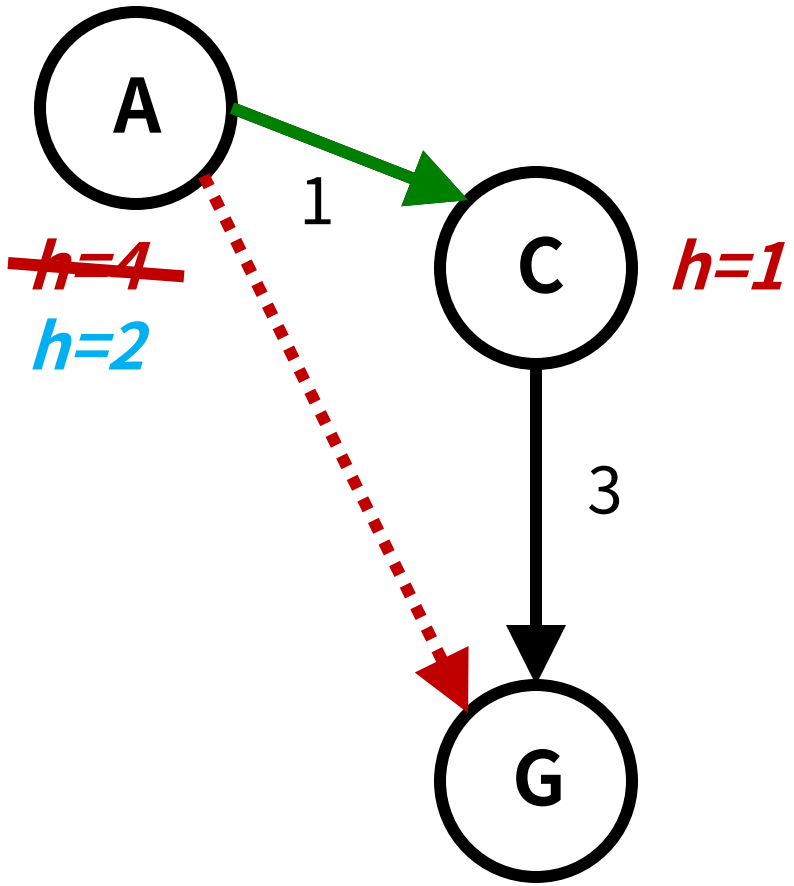
搜索树



闭集

{ SBCA }

启发式的一致性



§ 主要思想：估计启发式成本≤实际成本

§ 可接受性：启发式成本≤目标实际成本

哈）≤实际成本*H**从A到G

§ 一致性：启发式“弧”成本≤每个弧的实际成本

h(A) – h(C)≤成本（A 至 C）

§ 又称“三角不等式”：哈）≤成本（A 至 C）+ h(°C)

§ 注：真实成本*H**一定 满足三角不等式

§ 一致性的后果：

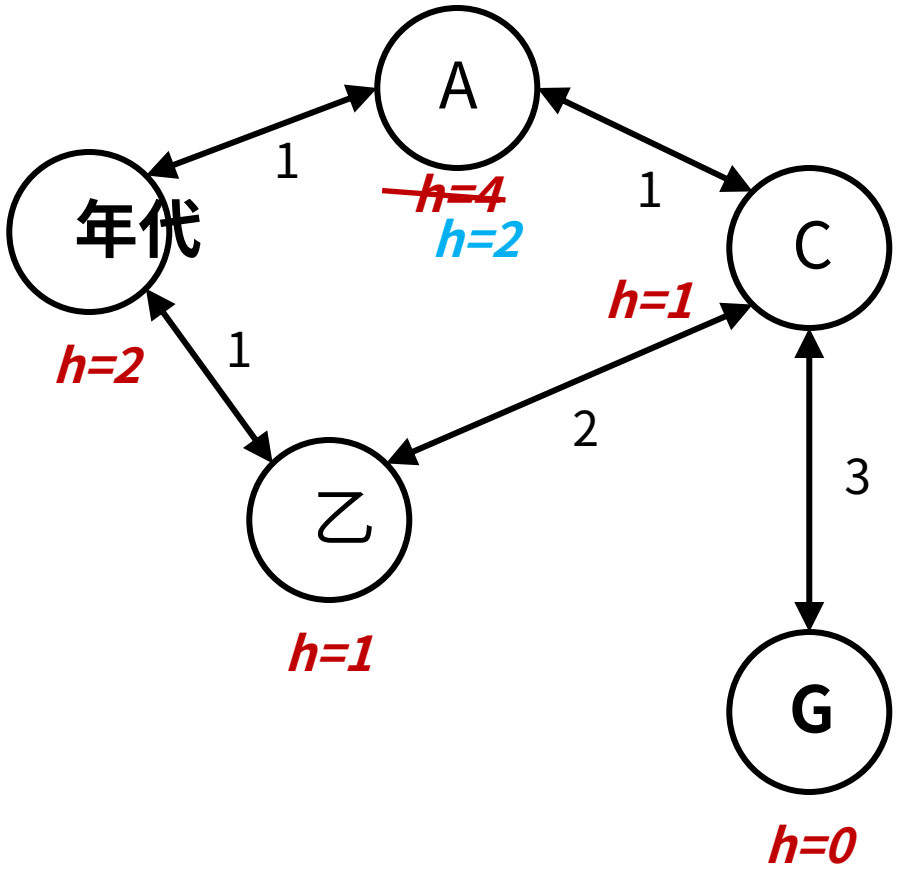
§ 沿路径的 f 值永远不会减小

哈）≤成本（A 至 C）+

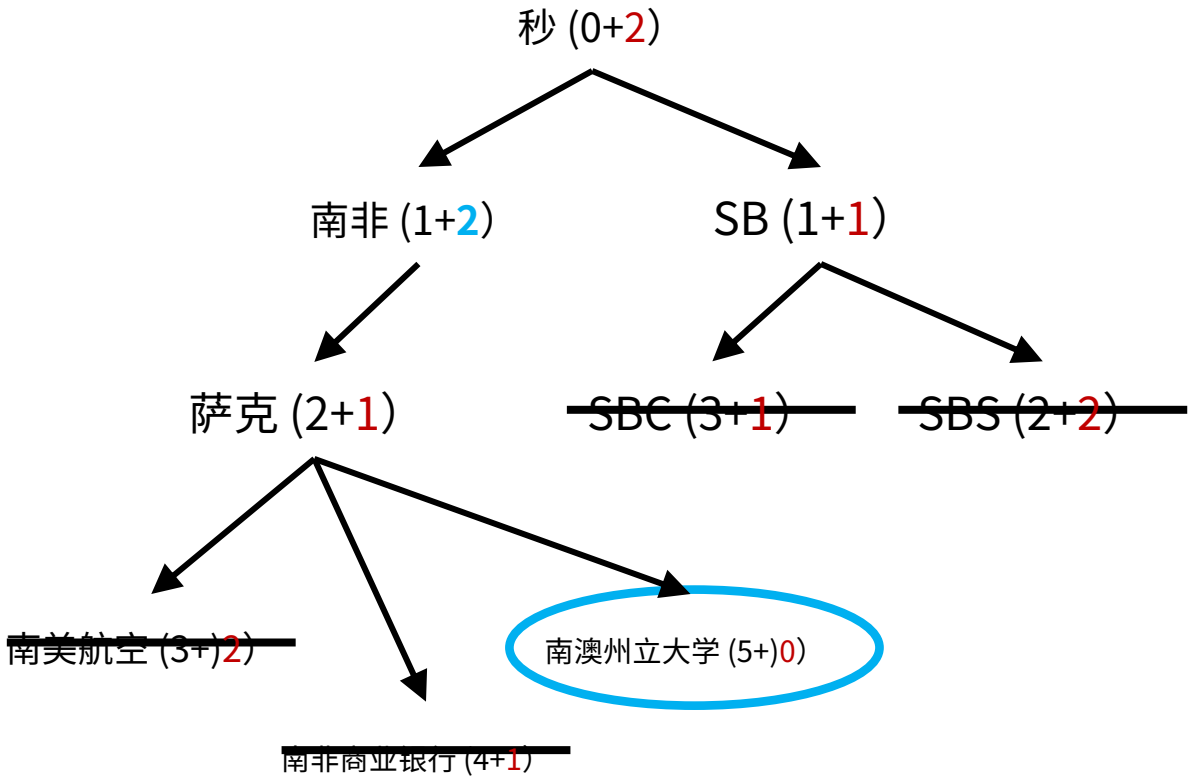
§ 温度 A* 图搜索是最优的

具有一致性启发式算法的 A* 图搜索

状态空间图



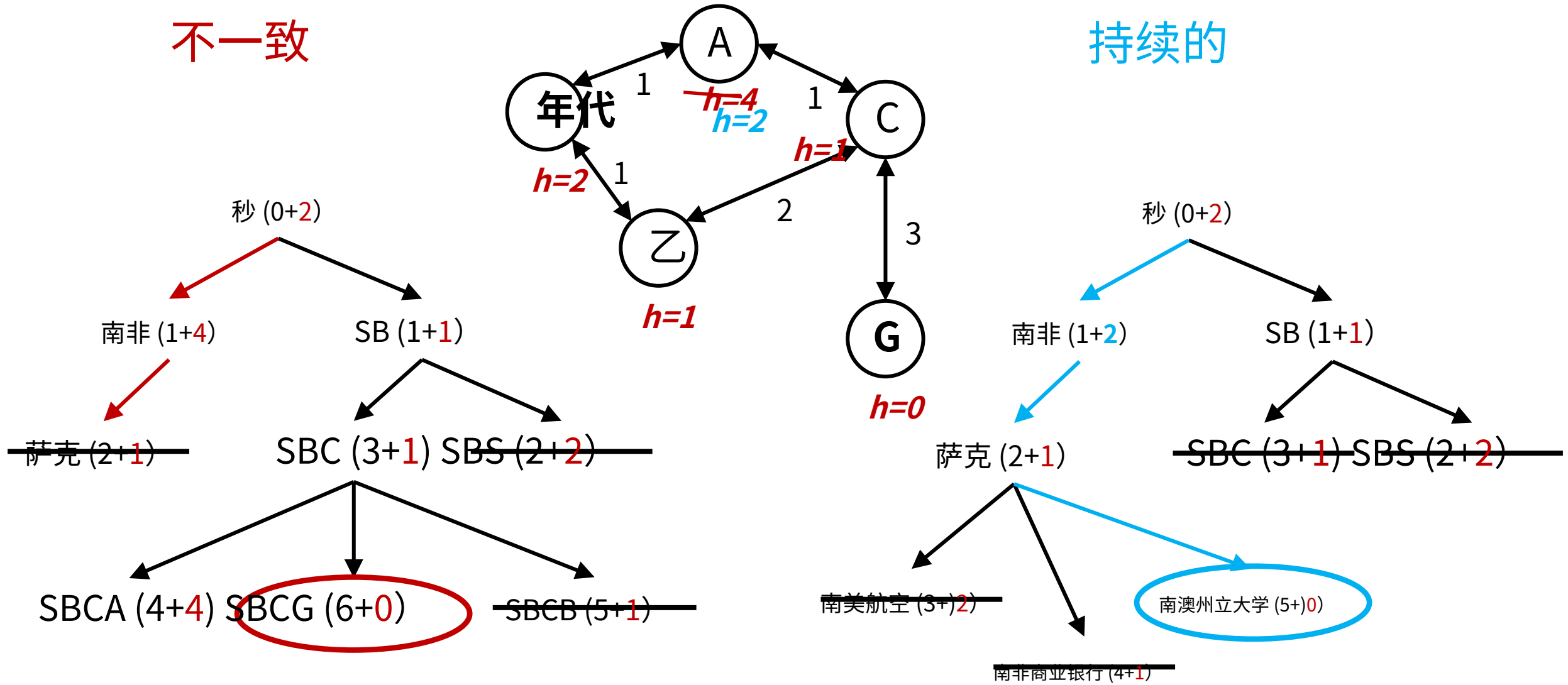
搜索树



闭集

{ SBAC }

一致性 => 非递减的 f 分数


不一致
A
年代
C
乙
G
1
1
1
2
3
h=4
h=2
h=2
h=1
h=1
h=0
秒 (0+2)
南非 (1+4)
SB (1+1)
萨克 (2+1)
SBC (3+1) SBS (2+2)
SBCA (4+4) SBCG (6+0)
SBCB (5+1)
持续的
秒 (0+2)
南非 (1+2)
SB (1+1)
萨克 (2+1)
SBC (3+1) SBS (2+2)
南美航空 (3+)2)
南澳州立大学 (5+)0)
南非商业银行 (4+1)

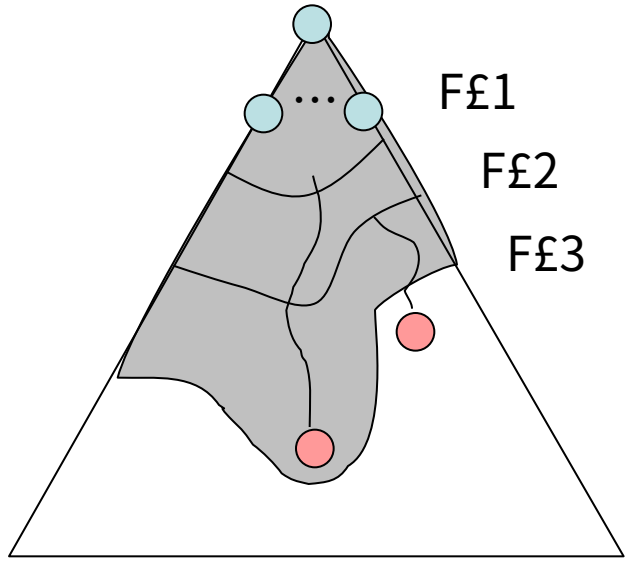
A* 图搜索的最优性

§ 概述：考虑一下 A* 如何处理
一致的启发式：

§ 事实 1：在树搜索中，A* 扩展节点
增加总 f 值（f 轮廓）

§ 事实 2：对于每个状态 s，到达
s 最优扩展，然后节点达到 s 次优

§ 结果：A* 图搜索是最优的



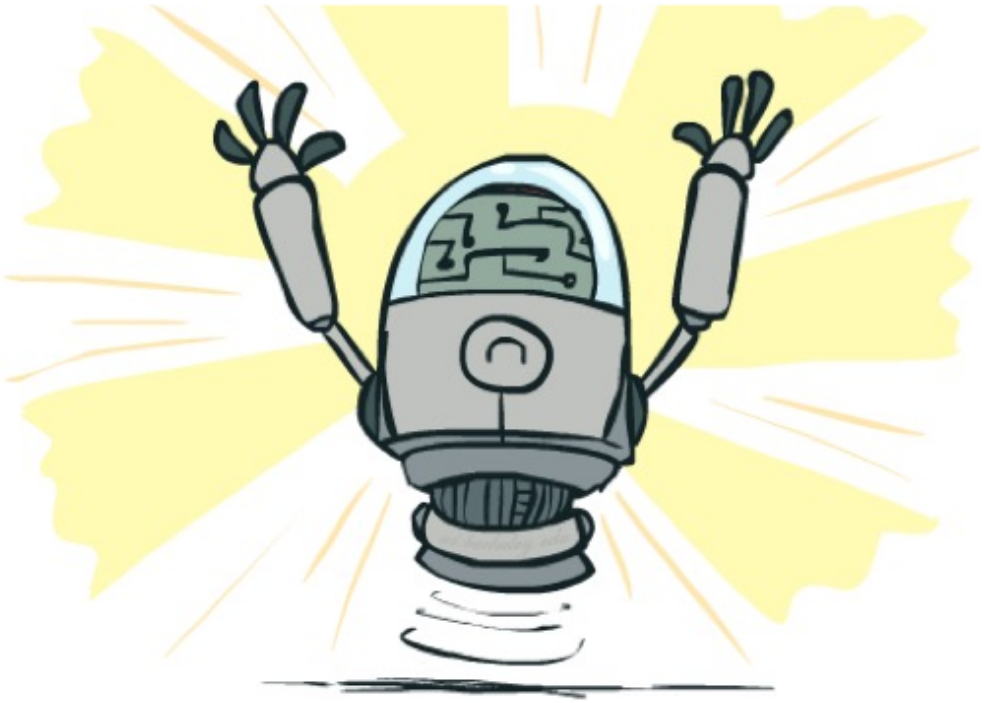
最优性

§ 树搜索：
  § 如果启发式方法可行，则 A* 是最优的
  § UCS 是一种特殊情况（h = 0）

§ 图表搜索：
  § 如果启发式算法一致，则 A* 最优
  § UCS 最优（h = 0 是一致的）

§ 一致性意味着可接受性

§ 一般来说，大多数自然可接受的启发式方法往往是一致的，特别是如果来自轻松的问题



但…

- A* 将整个探索区域保存在内存中
- =>在你无聊地等待答案之前就会用完空间☹
- 有一些使用较少内存的变体（第 3.5.5 节）：
  - IDA* 的工作原理类似于迭代深化，只不过它使用了一个*F*-limit 而不是深度限制
    - 在每次迭代中，记住最小的*F*-超过当前限制的值，用作新的限制
    - 效率很低*F*是实值，每个节点都有唯一值
  - RBFS 是一种递归深度优先搜索，它使用*F*-限制 =*F*- 最佳价值
    当前节点的任何祖先都提供替代路径
    - 当超出限制时，递归将展开，但会记住最佳可到达*F*-值
      那个分支
  - SMA* 用途*所有可用内存*对于队列，尽量减少抖动
    - 当队列满了的时候，删除队列中最差的节点，但记住它在父节点中的值

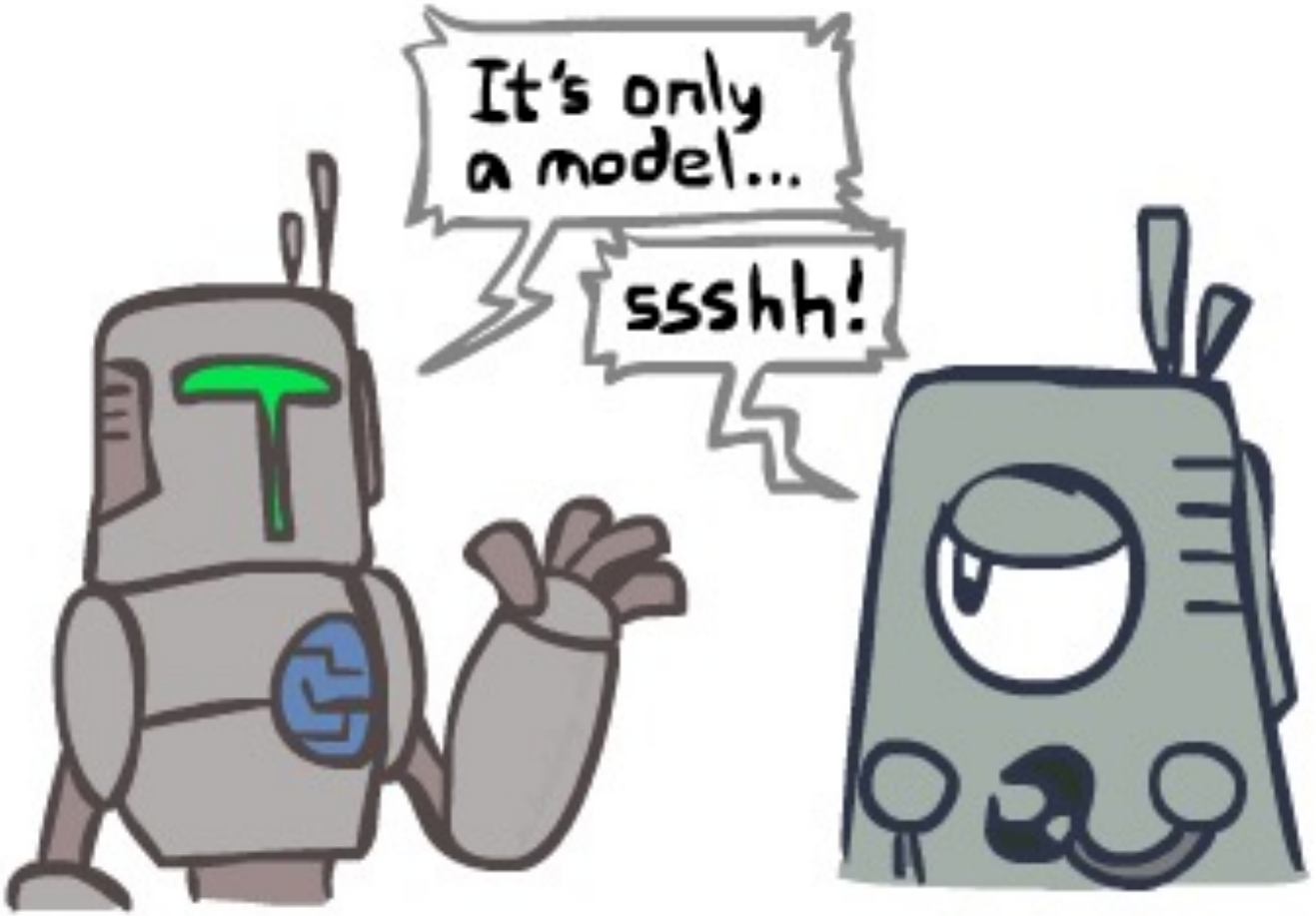
搜索和模型

§ 搜索操作
世界模型

§ 经纪人没有
在现实世界中真正尝试
所有计划！

§ 规划全在于
模拟”

§ 您的搜索仅作为
和你的模型一样好……



搜索 Gone W

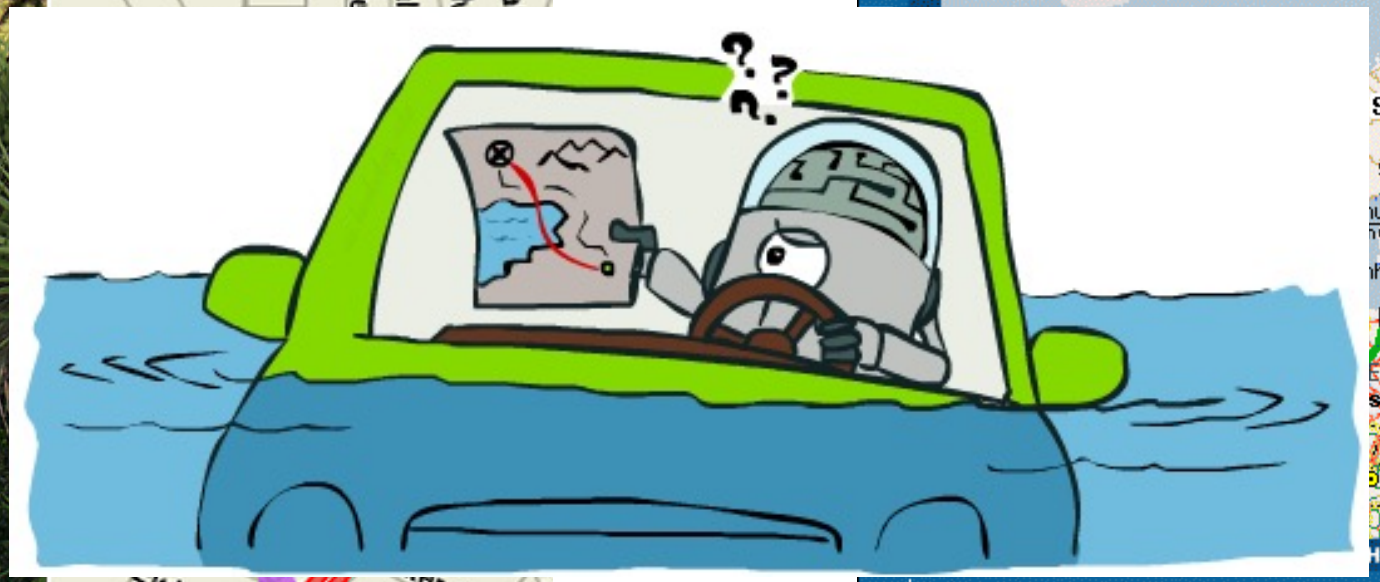

Loma Vista
2200 W
MAPQUEST
Brier Ave
Silver Lake Blvd
McCready Ave
Teviot St
Hidalgo Ave
Bancroft Ave
Silver Ridge Ave
Lake View Ave
Earl St
Loma Vista Pl
END
2
Cove Ave
Moore St
Oak Glen
Apex Ave
Glendale Blvd
Baxter St
Ewing St
2005 MapQuest.com, Inc.


NORTH
ARCTIC OCEAN
Microsoft MapPoint
ICELAND
End
ATLANTIC OCEAN
Start
SWEDEN
FINLAND
RUSSIA
Helsinki
Tver
Helsingfors
Stockholm
Riga
Smolensk
Växjö
Arhus
Aarhus
Edinburgh
Vilnius
Copenhagen
Białystok
BELARUS
IRE.
Kiev
Berlin
POLAND
Amsterdam
Wrocław
UKRAINE
GERMANY
London
Brussels
CZECH REP.
Chişinău
Paris
Bern
Berne
Vienna
HUNGARY
ROMANIA
Zagreb
FRANCE
Bucharest
©2004 Microsoft Corp ©2003 NAVTEQ
SOUTH
WEST
EAST
IN
km
mi
500
1000
200
400
600
Legend
Zoom on map cli
Start: Haugesund, Rogaland, Norway
End: Trondheim, Sør-Trøndelag, Norway
Total Distance: 2713.2 Kilometers
Estimated Total Time: 47 hours, 31 minutes
nrk.no/alltidmoro

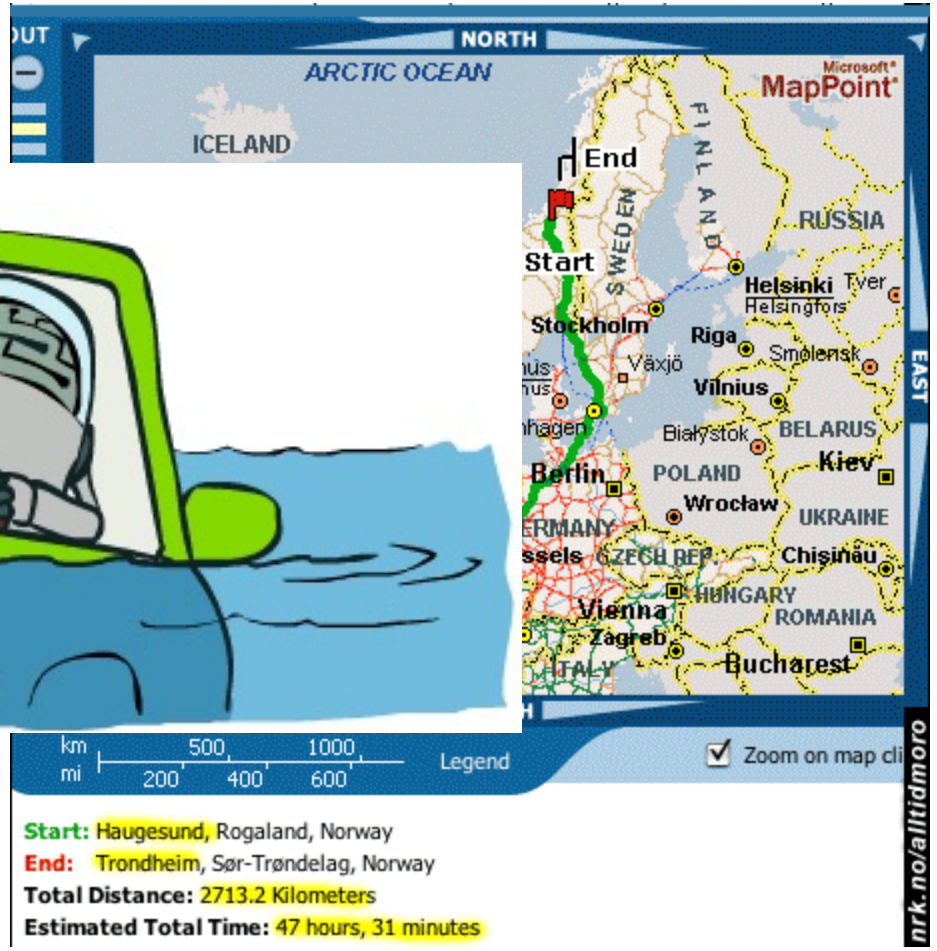
搜索 Gone W


Loma Vista
2200 W


MAPQUEST




Blvd
© 2005 MapQuest.com, Inc.


NORTH
ARCTIC OCEAN
ICELAND
Microsoft MapPoint
End
Start
SWEDEN
FINLAND
RUSSIA
Helsinki
Helsingfors
Tver
Stockholm
Riga
Växjö
Smolensk
Vilnius
Białystok
BELARUS
Kiev
Berlin
POLAND
Wrocław
UKRAINE
CZECH REP.
Chişinău
HUNGARY
Vienna
ROMANIA
Zagreb
Bucharest
km
mi
500
1000
200
400
600
Legend
Zoom on map cli
Start: Haugesund, Rogaland, Norway
End: Trondheim, Sør-Trøndelag, Norway
Total Distance: 2713.2 Kilometers
Estimated Total Time: 47 hours, 31 minutes
nrk.no/alltidmoro

树搜索伪代码

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        for child-node in EXPAND(STATE[node], problem) do
            fringe ← INSERT(child-node, fringe)
        end
    end
```

图搜索伪代码

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            for child-node in EXPAND(STATE[node], problem) do
                fringe ← INSERT(child-node, fringe)
            end
    end
```

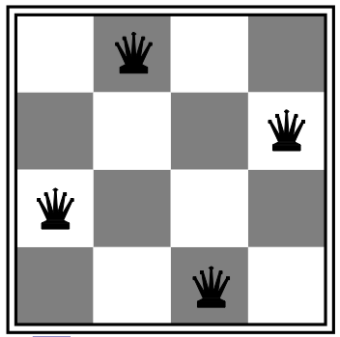
本地搜索

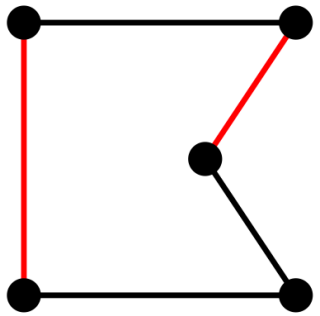
[这些幻灯片由 Dan Klein 和 Pieter Abbeel 为加州大学伯克利分校的 CS188 人工智能入门课程制作。所有 CS188 材料均可在 http://ai.berkeley.edu 获取。]
[更新幻灯片来自：Stuart Russell 和 Dawn Song]

本地搜索算法

§ 在许多优化问题中，***小路***无关紧要；目标状态***是***解决方案

§ 那么状态空间 = “完整”配置的集合；
寻找***满足约束的配置***例如，n 皇后问题；或者，找到 ***最优配置***例如，旅行商问题





§ 在这种情况下，可以使用***迭代改进***算法：保持单一“当前”状态，尝试改进它

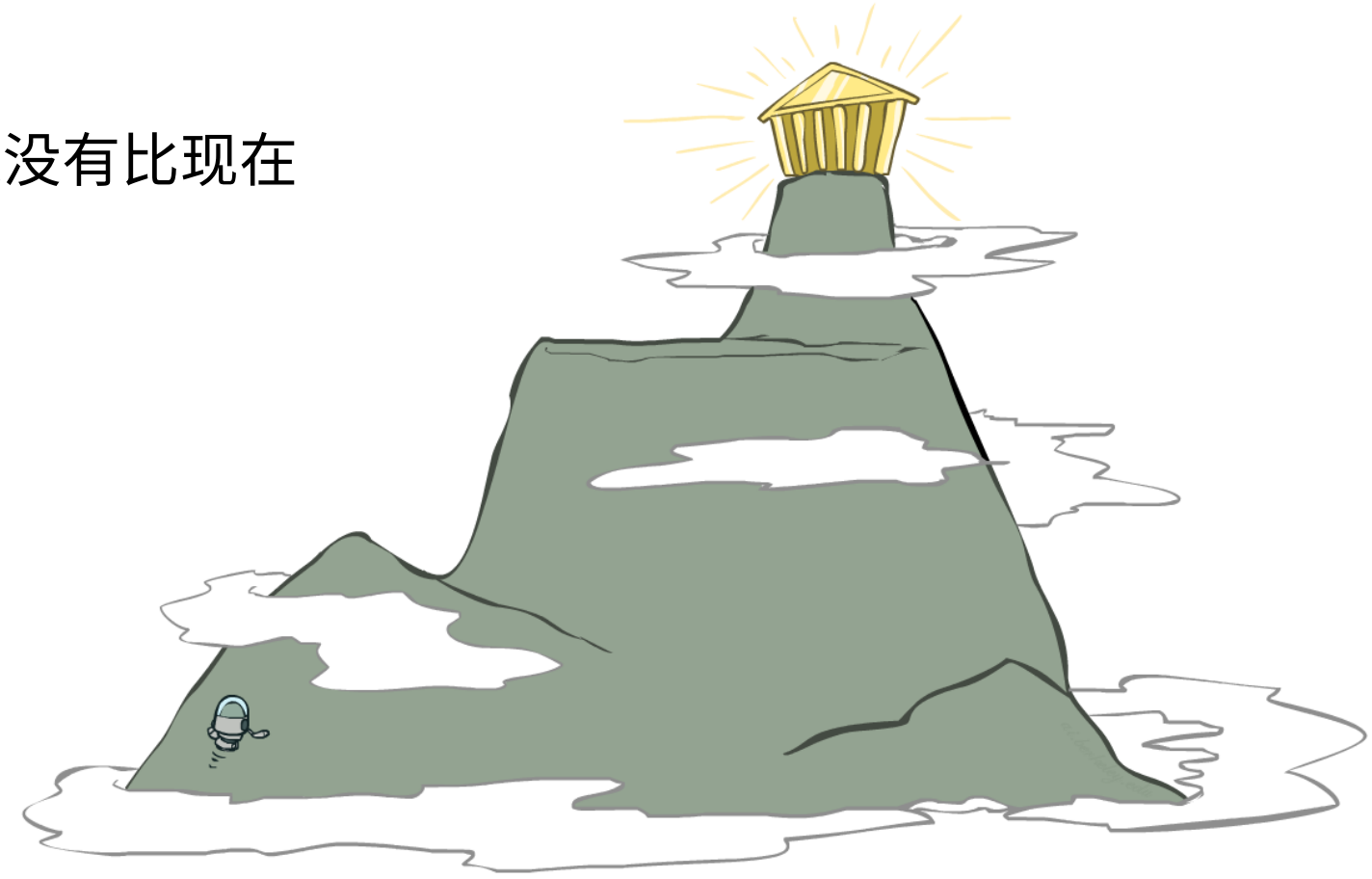
§ 恒定空间，适合在线和离线搜索 § 如果“状态”是你自己（即学习），这或多或少是不可避免的

爬山

§ 简单、总体的想法：

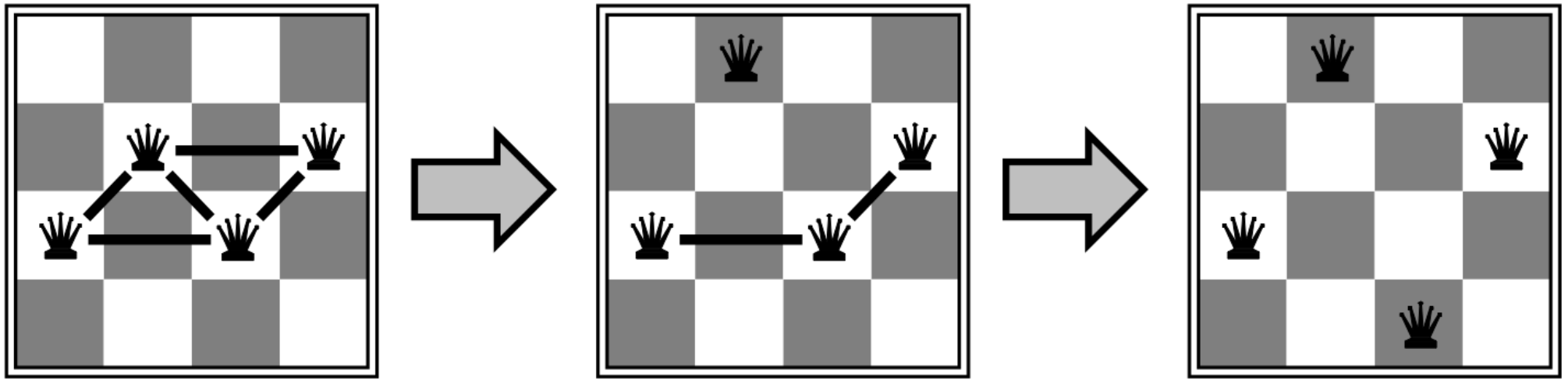
§ 从任意地点开始

§ 重复：移至最佳邻近州 § 如果没有比现在更好的邻居，就退出



HeurisUc 的*n*皇后问题

§ 目标：棋盘上有 n 个皇后，没有***冲突***即没有女王攻击另一个女王 § 状态：棋盘上有 n 个皇后，每列一个 § 动作：将皇后移动到其所在列 § 启发式价值函数：冲突次数

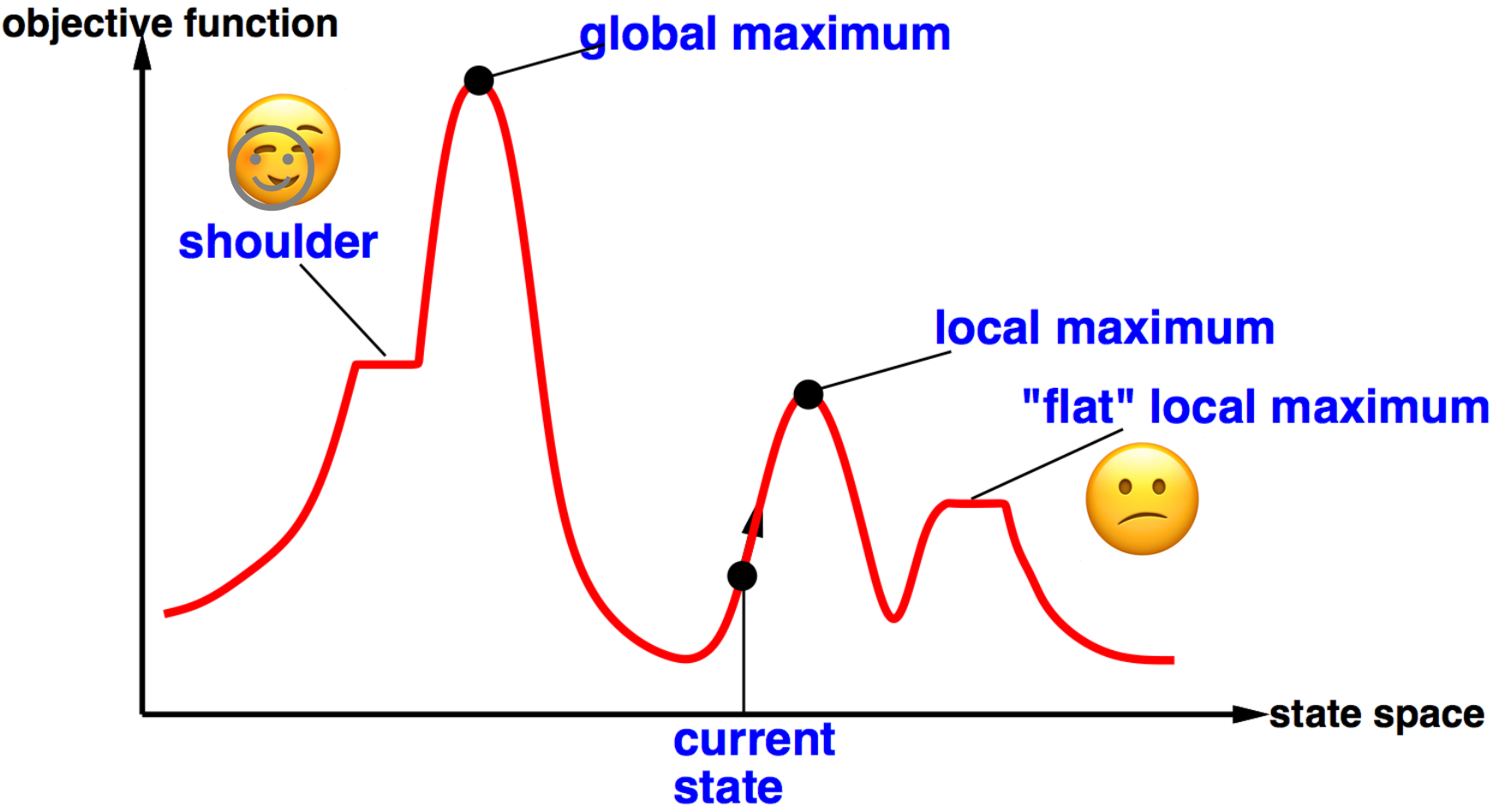


爬山算法

```
功能爬山（问题）返回一个状态
    当前的← make-node(问题.初始状态) 循环
    执行
        邻居← 最高价值的继任者当前的 如果邻居.值≤
        当前的。价值然后
            返回当前的。状态
        当前的←邻居
```

“就像在浓雾中失忆攀登珠穆朗玛峰一样”

全局和局部最大值



随机重启

§ 寻找全局最优 § 呃

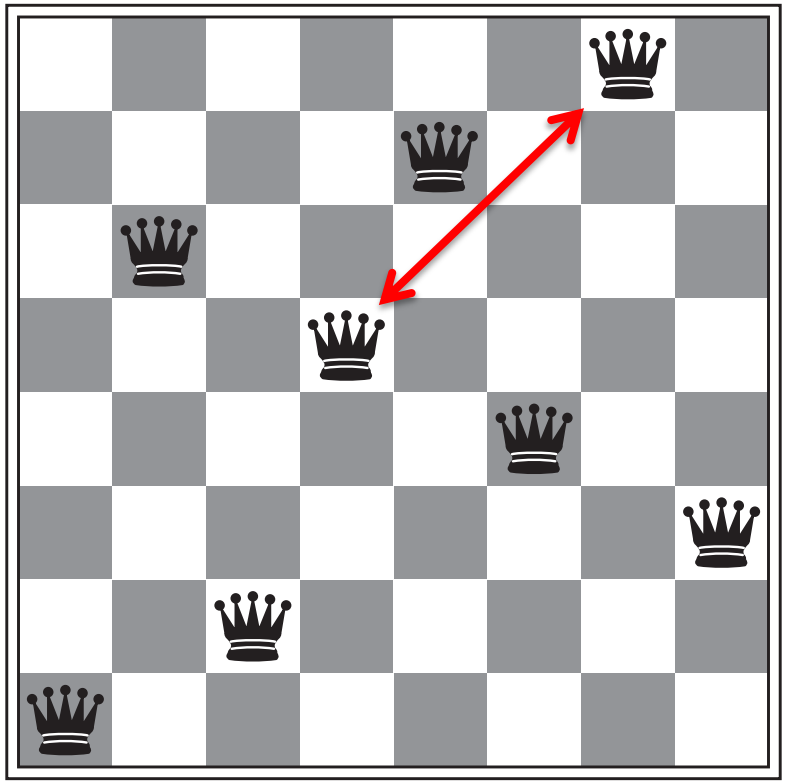
随机横向移动

§ 逃离肩膀 § 在平面上无限循环

局部最大值

八皇后问题的爬山法

- 无横向移动：
  - 成功概率为 0.14
  - 每次试验的平均移动次数：
    - 成功时为 4，遇到困难时为 3
  - 预计需要的总移动次数：
    - 3(1-p)/p + 4 =~ 22 步
- 允许 100 次横向移动：
  - 成功概率为 0.94
  - 每次试验的平均移动次数：
    - 成功时为 21，失败时为 65
  - 预计需要的总移动次数：
    - 65(1-p)/p + 21 =~ 25 步



寓意：需要调整的算法很烦人

模拟退火

§ 类似于退火工艺，将金属缓慢冷却至达到有序（低能量）状态

§ 基本思想：

§ 根据“温度”偶尔允许“坏”动作

§ 高温 => 允许更多坏动作，使系统脱离它的局部最小值

§ 按照一定的时间表逐渐降低温度 § 听起来很奇怪，不是吗?

模拟退火算法

功能模拟退火（问题，日程）**返回**一个状态 当前的←问题.初始状态

为了吨= 1**到**∞**做**

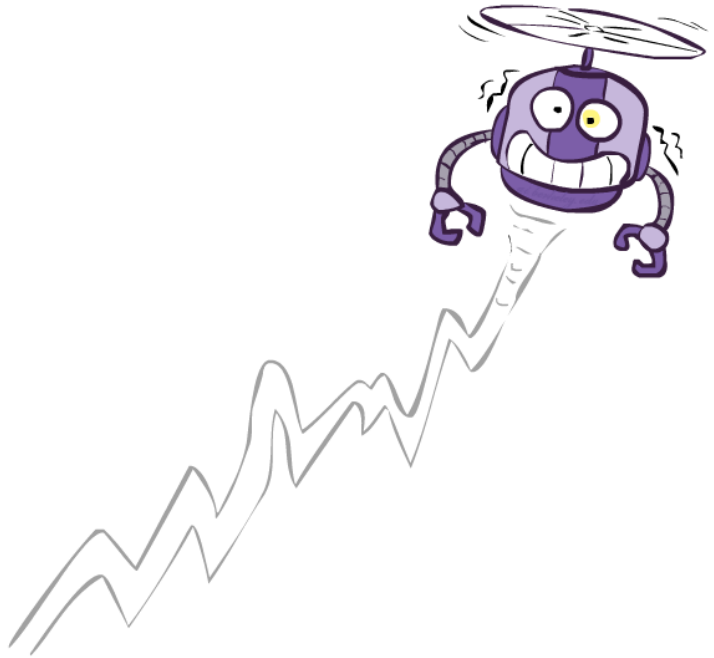
电视←日程（吨）

如果电视= 0**然后返回**当前的

下一个← 随机选择的继任者当前的

ΔE←下一个。价值 -当前的。价值 **如果**ΔE> 0**然后**当前的←下一个

别的当前的←下一个只有概率埃ΔE/T



模拟退火

§ 理论保证：

§ 平稳分布（玻尔兹曼）：*磷*（*X*）A*埃*埃（*X*)/*电视*

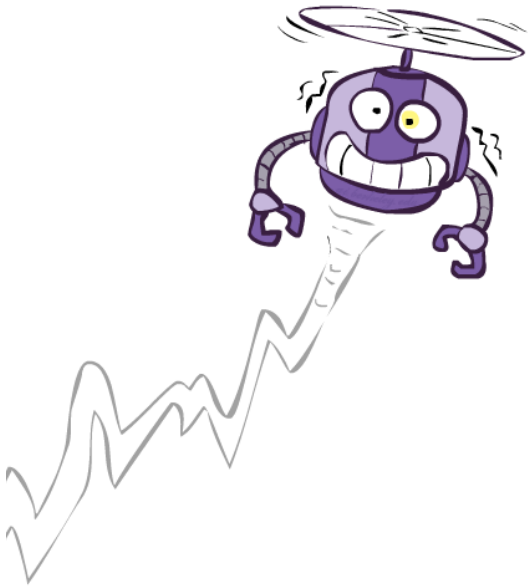
§ 如果*电视*下降得足够慢，就会收敛到最优状态！

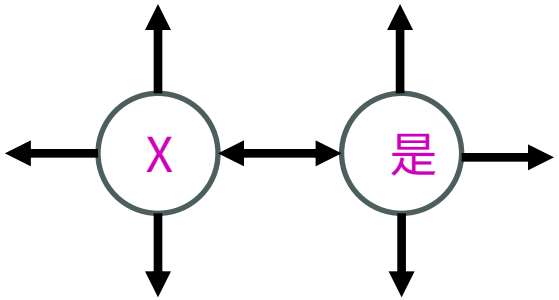
§ 证明草图

§ 考虑两个相邻的状态*X*，*是*和*埃*（*是*）>*埃*（*X*）[高即好]

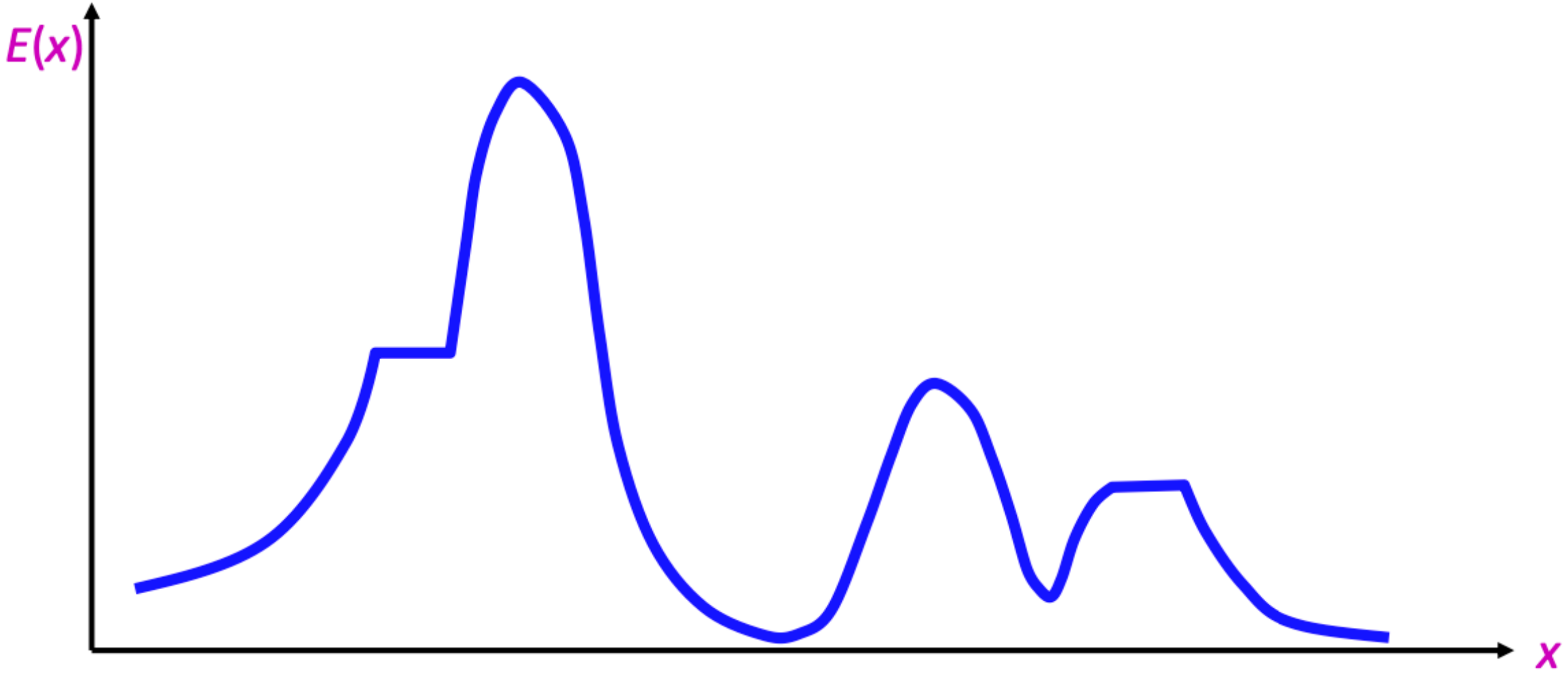
§ 认为*X*®*是*和*是*®*X*和出度*德*（*X*）=*德*（*是*）=*德* § 让*磷*（*X*），*磷*（*是*）是平衡占有概率*电视* § 让*磷*（*X*®*是*）是该状态的概率

*X*过渡到状态*是*





占用概率作为 T 的函数



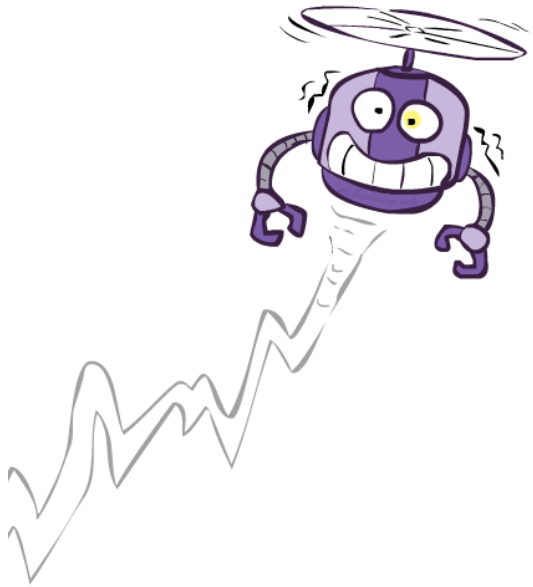
模拟退火

§ 这种融合是一种有趣的保证吗?

§ 听起来像魔术，但现实就是现实:

- § 你需要走下坡路的次数越多，才能摆脱局部最优，你连续完成所有任务的可能性就越小
- § “足够慢”可能意味着指数级缓慢
- § 随机重启爬山算法也收敛到最优状态……

§ 模拟退火及其相关算法是关键超大规模集成电路布局和其他最优配置问题中的主力



局部束搜索

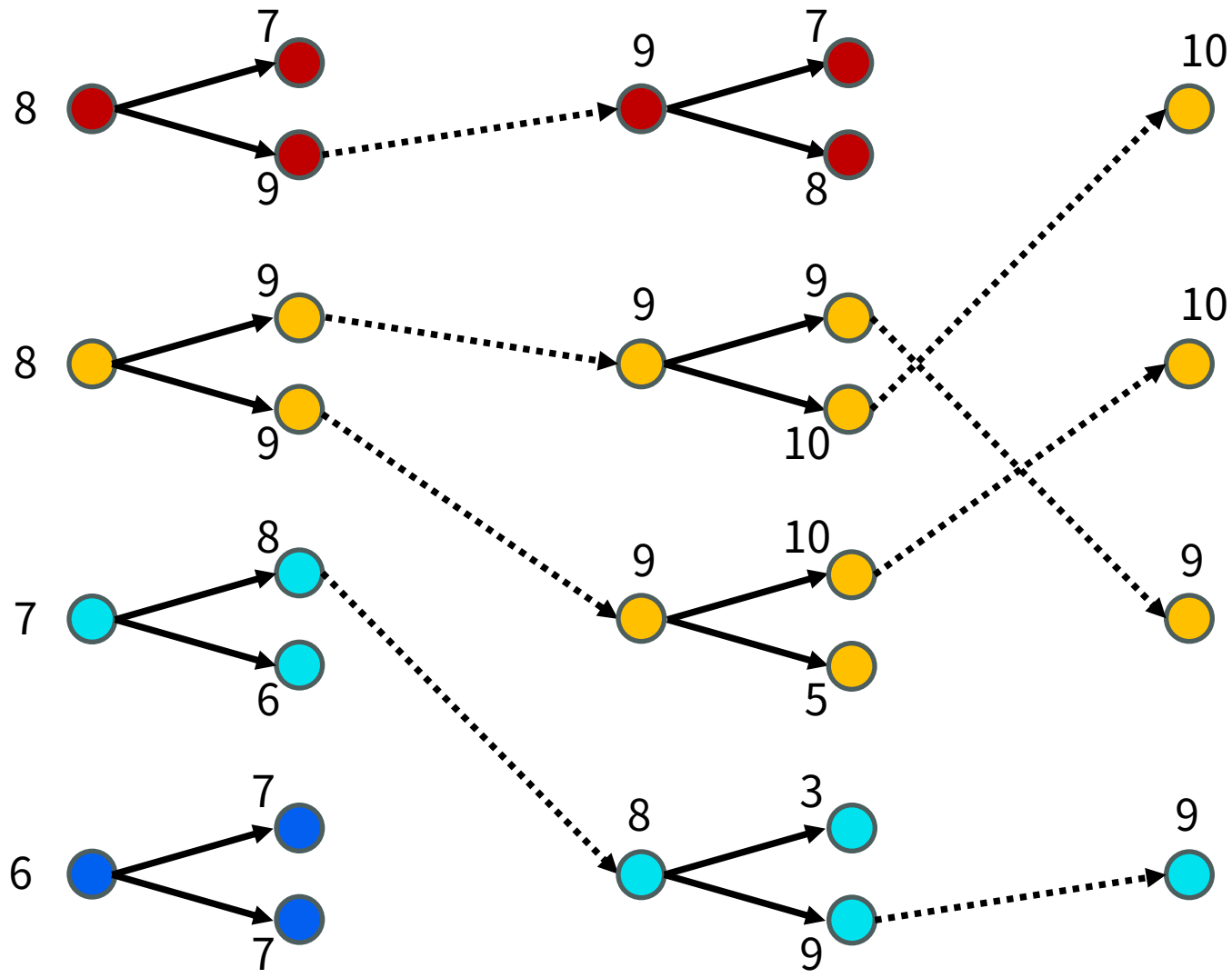
§ 基本思想：

§ *钾*本地搜索算法的副本，随机初始化 § 每次迭代

§ 生成所有后继者 *钾*当前状态 § 选择最好的这*钾*些是新的当前状态

或者，随机选择 K，偏向好的

集束搜索示例 (*钾=4*)



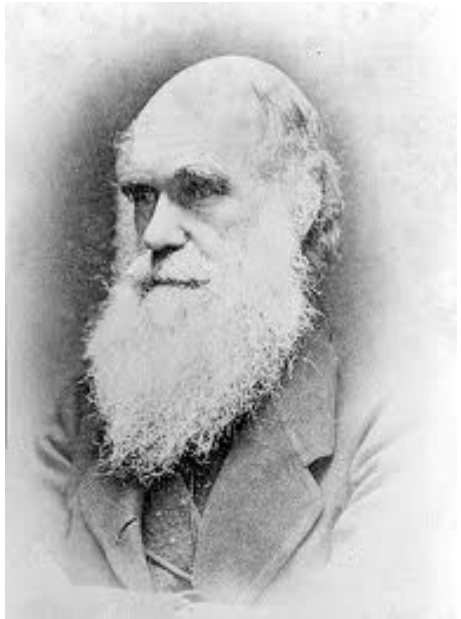
局部束搜索

§ 为什么这与 *钾*并行本地搜索?

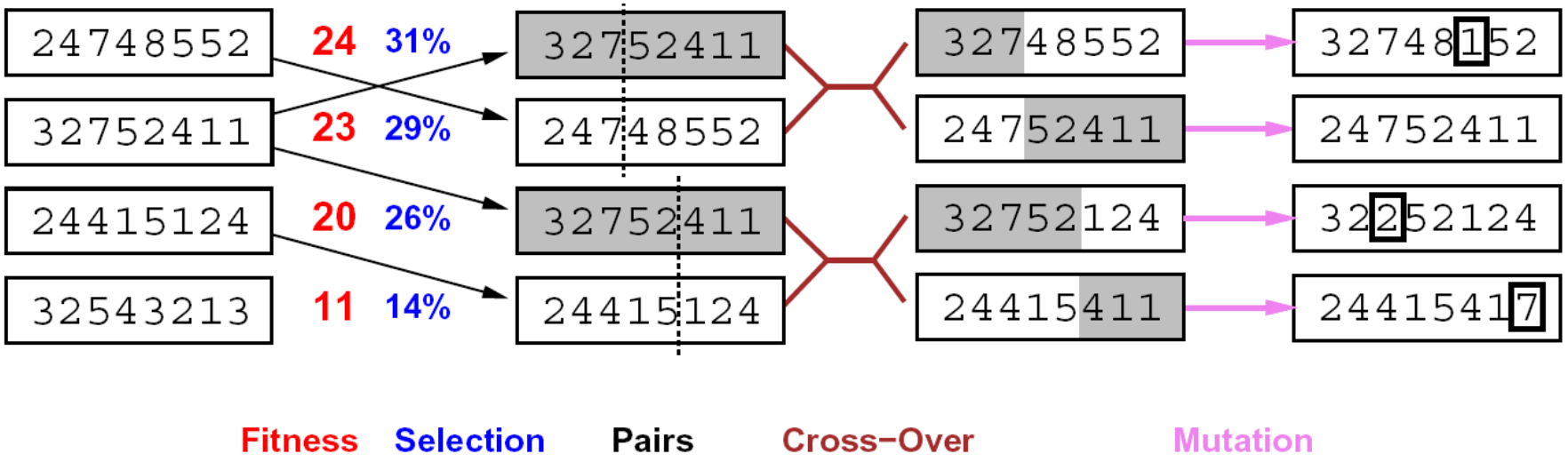
  § 搜索***交流***! “过来吧，这里的草更绿！”

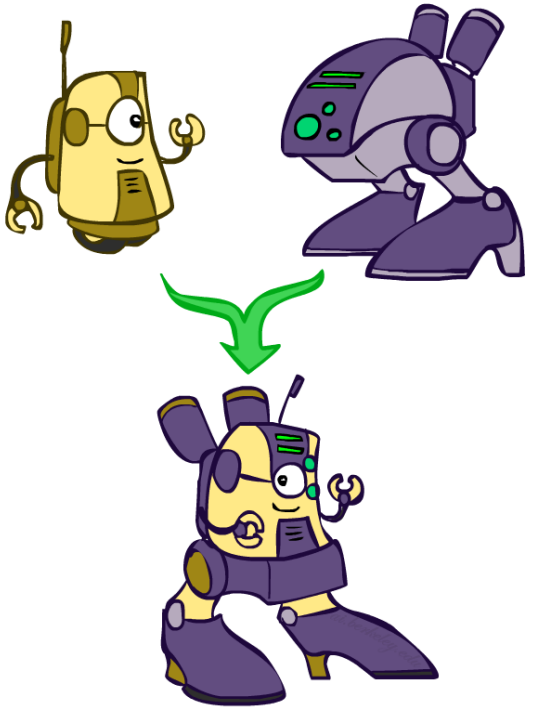
§ 这让你想起了哪个其他著名的算法?

  § 进化!



遗传算法

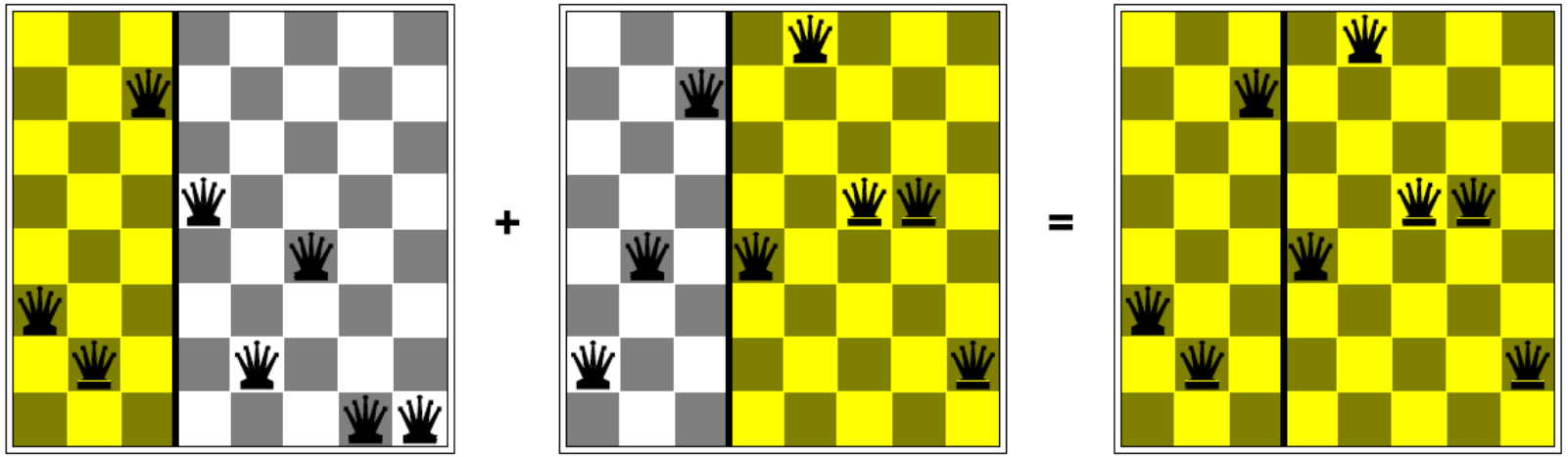




§ 遗传算法使用自然选择隐喻

§ 重新采样*钾*每一步（选择）的个体都由适应度函数加权 § 通过成对交叉算子结合，加上变异来提供多样性

例如：N-Queens



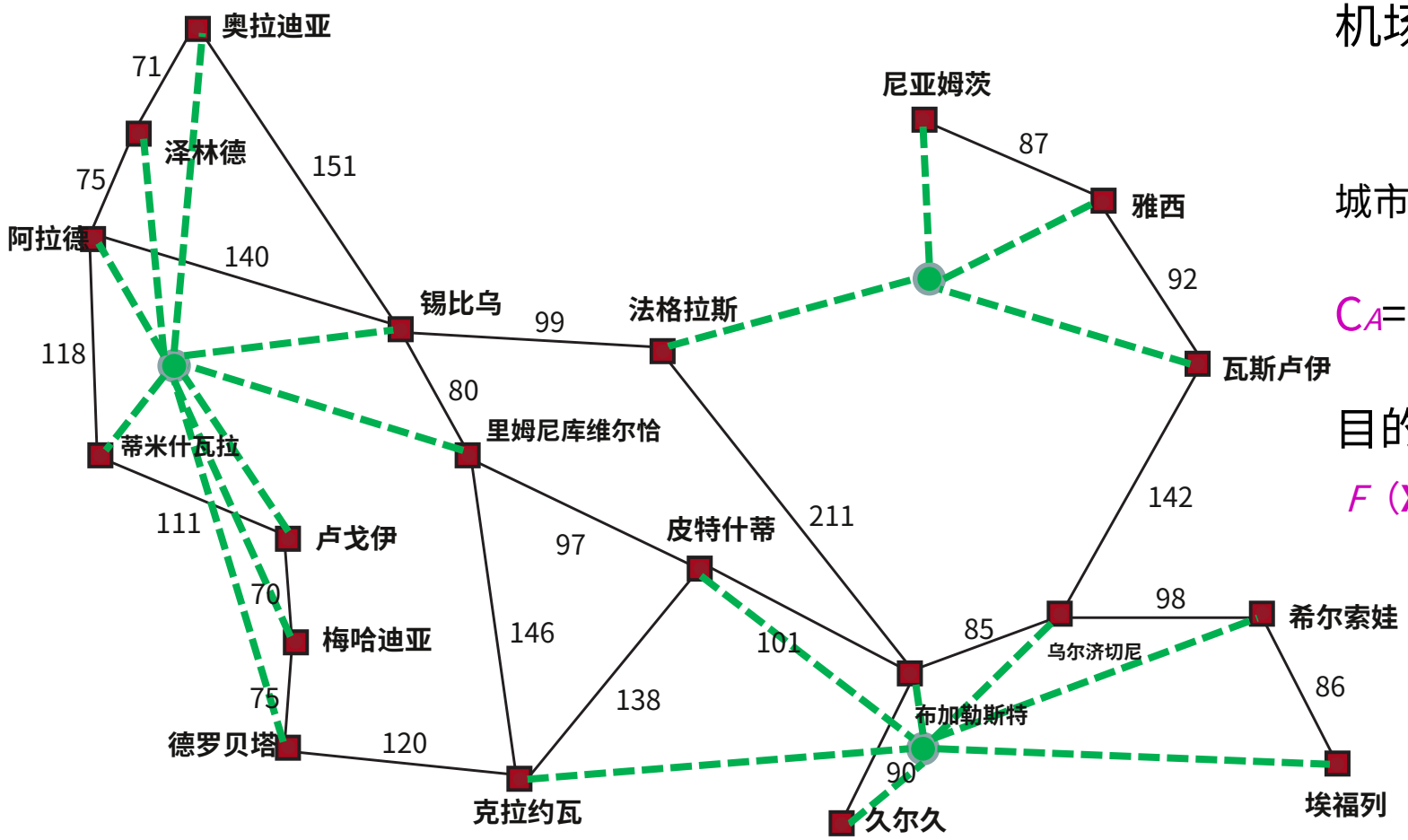
§ 这里的交叉有意义吗？ § 突变会是什么？

§ 良好的适应度函数是什么样的？

连续空间中的局部搜索

示例：在罗马尼亚放置机场

放置 3 个机场，以最小化每个城市到其最近机场的距离平方和



机场位置

$\mathbf{X} = (X_1, 是_1), (X_2, 是_2), (X_3, 是_3)$

城市位置 $(X_C, 是_C)$

C_A=距离机场最近的城市 A

目的：最小化

$$F(\mathbf{X}) = \text{年代}_A \text{年代}_{C_{我}} C (X_A - X_C)^2 + (是_A - 是_C)^2$$

A

处理连续状态/动作空间

1. 离散化！

   § 定义具有增量的网格d，使用任何离散算法

2. 选择对状态的随机扰动

   a. 首选爬山法：不断尝试，直到情况有所改善

   b.模拟退火

3. 计算梯度$F(\mathbf{X})$分析地

在连续空间中寻找极值

§ 渐变向量ÑF（**X**）=（¶ F/¶ X_1，¶ F/¶ 是$_1$，¶ F/¶ X_2，…） 电视

§ 对于机场来说，F（**X**）=年代$_A$年代$_C$我$_C$（X_A- X_C）^{2}+（是$_A$- 是$_C$）^{2}

§ ¶ F/¶ X_1=年代$_C$我$_C$2(X_1- X_C) §

在极值情况下，ÑF（**X**）= 0

§ 有时可以用封闭形式解决：X_1=（年代$_C$我$_C$$X_C$)/|Ç 1|

§ 这是局部最小值还是全局最小值F?

§ 如果我们不能解决ÑF（**X**）= 0以封闭形式…

§ 梯度下降：**X**¬**X**-一个F（**X**）

§ 利用梯度寻找极值的大量算法

概括

§ 许多配置和优化问题都可以制定为局部搜索

§ 常见的算法类型：

§ 爬山法，持续优化

§ 模拟退火（和其他随机方法） § 局部束搜索：多重交互搜索 § 遗传算法：打破和重组状态

许多机器学习算法都是本地搜索