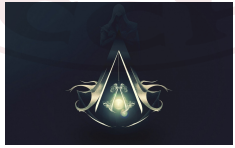
Search and Iteration Algortihm

laofu

陈江伦
Institute for Interdisciplinary Information Sciences
Tsinghua University

August 4, 2020



自我介绍？

参见去年课件“模拟费用流问题”

自我介绍？

参见去年课件“模拟费用流问题”
注意：去年的 Bonus 仍然没有被解决

Topics to talk about?

总而言之，有关优化的问题。

1. Search
   - Basic Search model
   - A* search
   - Adversarial search
   - Monte Carlo Tree Search(MCTS)

2. Iteration Algorithm
   - Zero-order Optimization
   - First-order Optimization
   - Graph Iteration

计算问题

- 优化
  - 可行解
  - 最优解
- 计数

What is Optimization?

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \text{constrants on } \mathbf{x} \end{array}$$

- 连续空间
- 离散空间
- 递归: Search
- 迭代: Do-While

What is search problem?

A search problem consists of:

- S : 状态空间
- S_{start} : 初始态
- Action(s) : 可行动作
- Succ(s, a) : 后继
- Cost(s, a) : 动作费用
- IsEnd(s) : 目标函数

一个 解 是一个从初始态出发到达一个终止态的可行动作序列

DFS

```
1: function DFS(s)
2:     if IsEnd(s) then
3:         return True
4:     end if
5:     for a ∈Action(s) do
6:         if DFS(Succ(s, a)) then
7:             return True
8:         end if
9:     end for
10:    return False
11: end function
```

DFS(non-recursive)

```
1: function DFS(s)
2:     St ← an empty stack
3:     St.push (s)
4:     if St is not empty then
5:         v ← St.pop()
6:         if IsEnd(v) then
7:             return True
8:         end if
9:         for a ∈Action(v) do
10:            St.push(Succ(v, a))
11:        end for
12:    end if
13:    return False
14: end function
```

BFS

```
1: function DFS(s)
2:     Q ← an empty queue
3:     Q.push (s)
4:     if Q is not empty then
5:         v ← Q.pop()
6:         if IsEnd(v) then
7:             return True
8:         end if
9:         for a ∈Action(v) do
10:            Q.push(Succ(v, a))
11:        end for
12:    end if
13:    return False
14: end function
```

A*

A*

- 用一个启发式函数 $h(x) = x$ 到达终点的估测距离

A*

- 用一个启发式函数 $h(x) = x$ 到达终点的估测距离
- 一个启发式函数 h 是 admissible(Optimistic) 当且仅当:

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

其中 $h^*(n)$ 是到达终点的真实距离

A*

- 用一个启发式函数 $h(x) = x$ 到达终点的估测距离
- 一个启发式函数 h 是 admissible(Optimistic) 当且仅当:

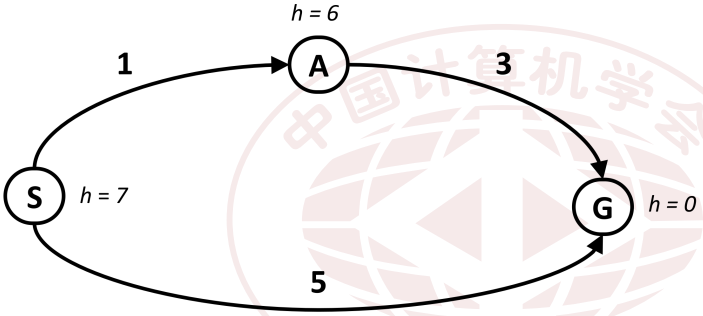
$$0 \leq h(n) \leq h^*(n)$$

其中 $h^*(n)$ 是到达终点的真实距离

- Consistency:

$$h(A) - h(C) \leq w_{a,c}$$

Is A* Optimal?



- 问题在哪?
- 真实距离 $<$ 估测距离
- 所以估测距离必须 $\leq$ 真实距离

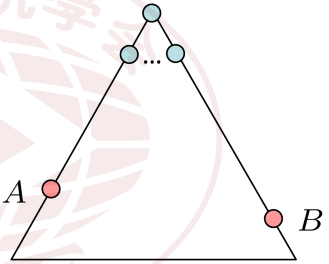
Optimality of A* Tree Search

假设:

- A 是最优解
- B 是一个非最优解
- h 满足 admissible

结论:

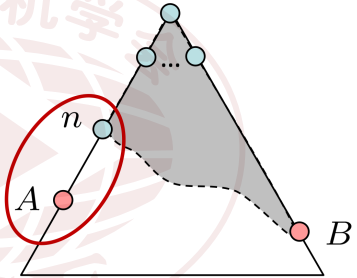
- A 将比 B 先访问到



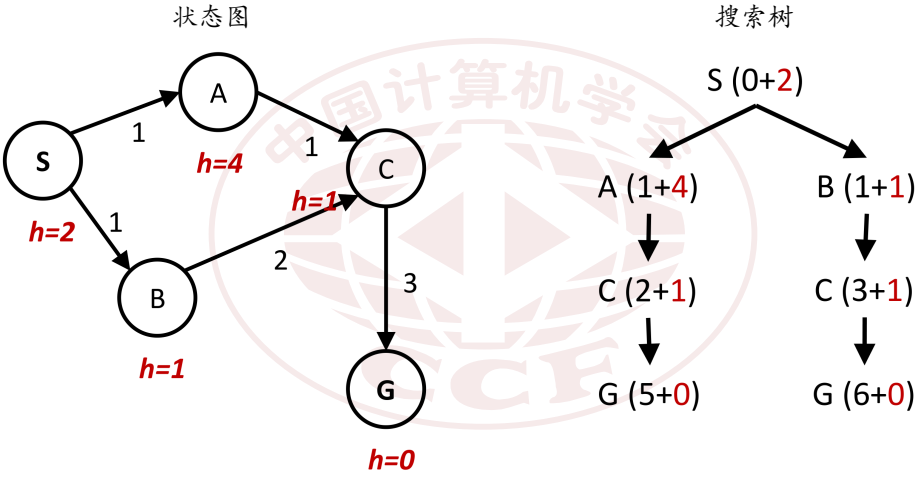
Optimality of A* Tree Search: Blocking

证明:

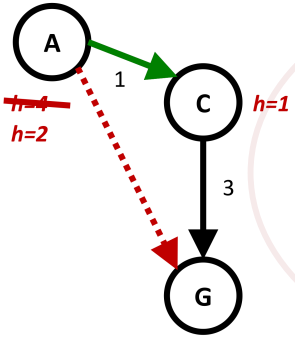
- n 为 A 的某一个祖先（可以是 A 自己）
- 结论: n 将比 B 先访问
- 令 $g(n)$ 为到达 n 已经走过的费用总和
- $f(n) = g(n) + h(n)$
- $f(n) \leq g(A)$: Admissibility
- $g(A) = f(A)$
- $g(A) < g(B)$
- $f(A) < f(B)$



How about A* in Graph Search?



Consistency of Heuristics



- 首先，估价函数必须 $\leq$ 最小费用

$$h(A) \leq w_{a,G}$$

- Consistency:

$$h(A) - h(C) \leq w_{A,C}$$

- 所以：

$$h(A) \leq w_{a,c} + h(C)$$

K-shortest path

Problem

找到 S 到 T 的第 k 短路

Algorithm

启发式函数: $h(x)$
一直迭代直到 x 被取出 x 次为止

IDA*

Problem

找到深度最浅的节点

DFS? 节省空间，但是无法限制搜索深度
BFS? 按深度搜索，但是消耗过多的空间
两者结合: IDA*

IDA*

Algorithm

从低到高枚举答案
　按照 DFS 序进行搜索.

Adversarial search

MinMax Values

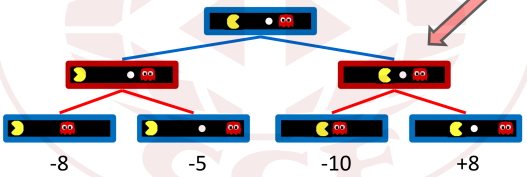
Description

States Under Agent's Control:

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

States Under Opponent's Control:

$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$



Terminal States:

$$V(s) = \text{known}$$

MinMax Search

```
1: function MAX-VALUE(state)
2:     v ← −∞
3:     for each successor of state do
4:         v ← max(v,Min-Value(succ))
5:     end for
6:     return v
7: end function
```

```
1: function MIN-VALUE(state)
2:     v ← ∞
3:     for each successor of state do
4:         v ← min(v,Max-Value(succ))
5:     end for
6:     return v
7: end function
```

MinMax Search

```
1: function MAX-VALUE(state)
2:     v ← −∞
3:     for each successor of state do
4:         v ← max(v,Min-Value(succ))
5:     end for
6:     return v
7: end function
```

```
1: function MIN-VALUE(state)
2:     v ← ∞
3:     for each successor of state do
4:         v ← min(v,Max-Value(succ))
5:     end for
6:     return v
7: end function
```

时间复杂度 $O(b^m)$
空间复杂度 $O(m)$

Alpha-Beta Pruning

```
1:  function MAX-VALUE(state,α,β)
2:      v ← −∞
3:      for each successor of state do
4:          v ← max(v,Min-Value(succ,α,β))
5:          if v ≥ β then
6:              return v
7:          end if
8:          α ← max(α, v)
9:      end for
10:     return v
11: end function
```

```
1:  function MIN-VALUE(state,α,β)
2:      v ← ∞
3:      for each successor of state do
4:          v ← min(v,Max-Value(succ,α,β))
5:          if v ≤ α then
6:              return v
7:          end if
8:          β ← min(β, v)
9:      end for
10:     return v
11: end function
```

Alpha-Beta Pruning

```
1: function MAX-VALUE(state,α,β)
2:     v ← −∞
3:     for each successor of state do
4:         v ← max(v,Min-Value(succ,α,β))
5:         if v ≥ β then
6:             return v
7:         end if
8:         α ← max(α, v)
9:     end for
10:    return v
11: end function
```

```
1: function MIN-VALUE(state,α,β)
2:     v ← ∞
3:     for each successor of state do
4:         v ← min(v,Max-Value(succ,α,β))
5:         if v ≤ α then
6:             return v
7:         end if
8:         β ← min(β, v)
9:     end for
10:    return v
11: end function
```

最优时间复杂度?

Alpha-Beta Pruning

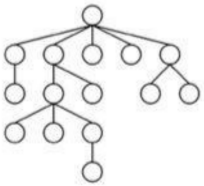
```
1: function MAX-VALUE(state,α,β)
2:     v ← −∞
3:     for each successor of state do
4:         v ← max(v,Min-Value(succ,α,β))
5:         if v ≥ β then
6:             return v
7:         end if
8:         α ← max(α, v)
9:     end for
10:    return v
11: end function
```

```
1: function MIN-VALUE(state,α,β)
2:     v ← ∞
3:     for each successor of state do
4:         v ← min(v,Max-Value(succ,α,β))
5:         if v ≤ α then
6:             return v
7:         end if
8:         β ← min(β, v)
9:     end for
10:    return v
11: end function
```

最优时间复杂度? $O(b^{m/2})$

MCTS Overview

- 在树上采样搜索
- 迭代
  - 找一条采样路径
  - 更新权值函数



Meeting the Middle

Problem

有 n 个数，要求你找到一个和为 0 的非空子集。
$n \leq 40$

Meeting the Middle

NOI2001 方程的解数

已知 $1 \leq x_i \leq m$，求方程 $\sum_{i=1}^{n} k_i x_i^{p_i} = 0$ 有多少组整数解

$n \leq 6, m \leq 160, |\sum_{k_i} m^{p_i}| < 2^{31}$

Dancing Links

- 精确覆盖问题
- 一个 $n \times m$ 的 01 矩阵，要求选出一个行的子集，使得在这些选出的行中，每一列都恰好有一个 1。
- 数据结构：双向十字链表
- 固定一列，枚举选择使用哪一行
- 把这一列所有行全部删除

Iteration

搜索：一般定义在树/图空间上
迭代：可以定义在树/图空间上，也可以定义在欧式空间/离散欧式空间上

爬山

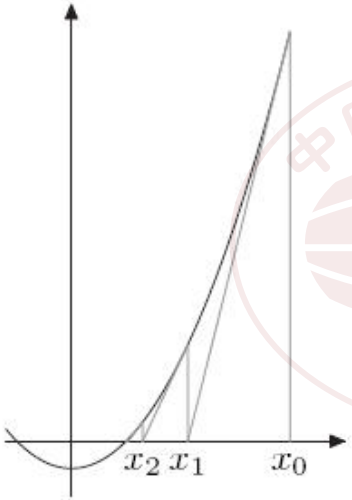
$$x_{n+1} = \underset{x' \in \text{Neighbor}(x_n)}{\text{argmin}} \quad (f(x'))$$

从一个点出发，每次按照最近的邻居走

模拟退火

$$P(x_n \to x') = \begin{cases} 1 & f(x') < f(x_n) \\ e^{\frac{f(n)-f(x')}{t}} & f(x') \geq f(x_n) \end{cases}$$

Newton-Raphson method



$$y = f'(x_n)(x - x_n) + f(x_n)$$

$$0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Failure Analysis

错误的初始点

$$f(x) = 1 - x^2$$

$$x_0 = 0,\ f'(0) = 0.$$

循环的初始点

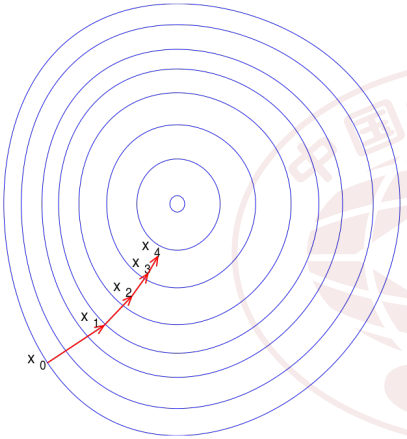
$$f(x) = x^3 - 2x + 2$$

Start from $x_0 = 0$.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = -\frac{2}{-2} = 1$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1 - \frac{1}{1} = 0$$

Gradient descent(GD)



$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla F(\mathbf{x}_n)$$

Graph Iteration

赌徒问题

有一个赌徒，初始有 k 元，每次他有 p 的概率赢一元钱，有 $1-p$ 的概率输一元钱，当他赢到 m 元时就会离开，当手上没有钱时就会破产，问赌徒一直玩下去离开的概率

Graph Iteration

赌徒问题

有一个赌徒，初始有 k 元，每次他有 p 的概率赢一元钱，有 $1-p$ 的概率输一元钱，当他赢到 m 元时就会离开，当手上没有钱时就会破产，问赌徒一直玩下去离开的概率

迭代做法：
随机一个函数 $f_0 : [m] \to [0,1]$
$\forall n \in \mathbb{N}^+, f_n(x) = pf_n(x-1) + (1-p)f_n(x+1)$

Graph Iteration

赌徒问题 2

有一个赌徒，初始有 k 元，每次他可以押注 t 元，然后有 p 的概率再赢 t 元钱，有 $1-p$ 的概率输掉这 t 元，当他赢到 $\geq m$ 元时就会离开，当手上没有钱时就会破产，问赌徒一直玩下去离开的概率

Graph Iteration

赌徒问题 2

有一个赌徒，初始有 k 元，每次他可以押注 t 元，然后有 p 的概率再赢 t 元钱，有 $1-p$ 的概率输掉这 t 元，当他赢到 $\geq m$ 元时就会离开，当手上没有钱时就会破产，问赌徒一直玩下去离开的概率

迭代做法：
随机一个函数 $f_0 : [m] \to [0, 1]$
$\forall n \in \mathbb{N}^+, f_n(x) = \max_{y \in [1,x]} pf_n(x-y) + (1-p)f_n(x+y)$

MDP Problem

MDP 问题：

- 状态集 S
- 可行动作 A
- 转移函数 $P(s, a, s') \to [0, 1]$
- 收益函数 $R(s)$
- 衰减系数 $\gamma \in (0, 1]$

如果一个样本的状态序列为 $s_0, s_2, \ldots, s_n$，则总收益为 $\sum_{i=0}^{n} R(s_i)\gamma^i$

Topological Graph?

动态规划问题

$V(s) = R(s) + \max_{a \in A} \sum_{s' \in S} \gamma P(s, a, s') V(s)$

General Graph?

迭代做法：
随机一个初始分布 V_0
$\forall n \in \mathbb{N}^+ V_n(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P(s, a, s') V_{n-1}(s)$

收敛性

$$
\begin{aligned}
& V_{n+1}(s) - V_n(s) \\
=& \gamma |\max_{a \in A} \sum_{s' \in S} P(s,a,s') V_n(s') - \max_{a \in A} \sum_{s' \in S} P(s,a,s') V_{n-1}(s')| \\
\leq & \gamma \max_{a \in A} |\sum_{s' \in S} P(s,a,s') V_n(s') - \sum_{s' \in S} P(s,a,s') V_{n-1}(s')| \\
\leq & \gamma \max_{s \in S} |V_n(s) - V_{n-1}(s)|
\end{aligned}
$$

Policy Iteration

另一种迭代做法：
初始一个策略 π_0 和权值 V_0
$\forall n \in \mathbb{N}^+$:

$$\pi_n(s) = \text{argmax}_{a \in A} \gamma \sum_{s' \in S} P(s, a, s') V_{n-1}(s)$$

$$V_n(s) = R(s) + \gamma \sum_{s' \in S} P(s, \pi_n(s), s') V_{n-1}(s)$$

HNOI2013

一场足球赛共 N 支球队参加，比赛规则如下：

- 每两只球队之间踢一场比赛
- 平局各得一分
- 胜利得三分，败者不得分

给定每个球队的最终得分，求有多少种可能的比赛结果，$N \leq 10$

Thanks

Thanks

- €€£?(Awesome salary)

GL & HF

记得去年的 Bonus 哦!