# 多智能体

# HW2

201300035 方盛俊 人工智能学院

# 目录

# 课后作业 4-1

## (1)

纯策略纳什均衡解: (D, D)

帕累托最优解: (C, D), (D, C), (C, C)

社会福利最优解: (C, C)

## (2)

纯策略纳什均衡解: (C, D), (D, C)

帕累托最优解: (C, D), (D, C), (C, C)

社会福利最优解: (C, C)

## (3)

纯策略纳什均衡解: (C, D), (D, C)

帕累托最优解: (C, D), (D, C)

社会福利最优解: (C, D), (D, C)

# 课后作业 4-2

## (1)

纯策略纳什均衡解: (C, D), (D, C)

帕累托最优解: (C, D), (D, C), (C, C)

社会福利最优解: (C, D), (D, C), (C, C)

## (2)

纯策略纳什均衡解: 无

帕累托最优解: (D, D), (C, D), (D, C), (C, C)

社会福利最优解: (D, D), (C, D), (D, C), (C, C)

## (3)

纯策略纳什均衡解: (D, D)

帕累托最优解: (D, D)

社会福利最优解: (D, D)

# 课后作业 4-3

需要看双方程序采取了什么策略, 例如如果双方程序选择的策略都是如果双方程序的代码相同, 则进行合作的话, 则就达成了相互合作的程序均衡解.

深究其原因的话, 是因为可以获取到对方程序的代码, 进而得知获取到对方的出价策略, 以此来选择自己的策略. 因为可以 使用可证明性逻辑来构建代理程序, 使得可以以一种稳健的方式实现相互合作, 这种合作甚至不需要代理程序的源代码完全相等, 也可以实现不会被利用的相互合作程序均衡解.

# 课后作业 4-4

**多数制:** comedy.

**波达计数:**

计算四种电影类型的最终奖励:

$\text{action} = 3 \times 3 + 2 \times 0 + 5 \times 2 + 3 \times 1 = 22$

$\text{comedy} = 3 \times 1 + 2 \times 1 + 5 \times 3 + 3 \times 0 = 20$

$\text{drama} = 3 \times 2 + 2 \times 2 + 5 \times 1 + 3 \times 3 = 24$

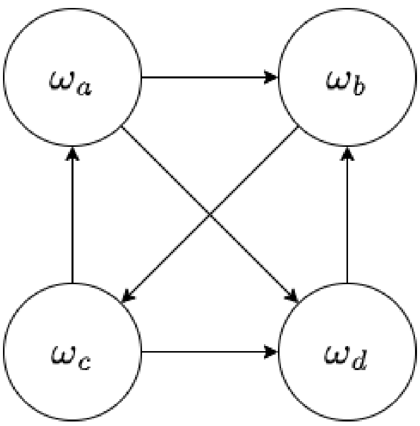
$\text{romance} = 3 \times 0 + 2 \times 3 + 5 \times 0 + 3 \times 2 = 12$

因此波达计数结果为 drama.

# 课后作业 4-5

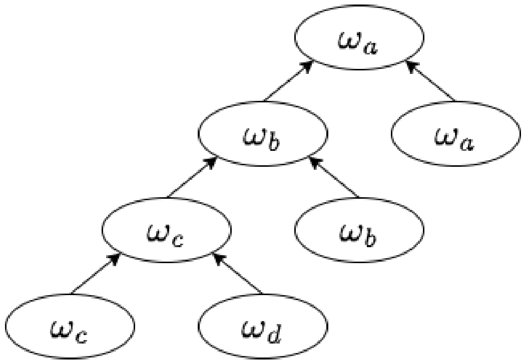
## (1)

多数图为:



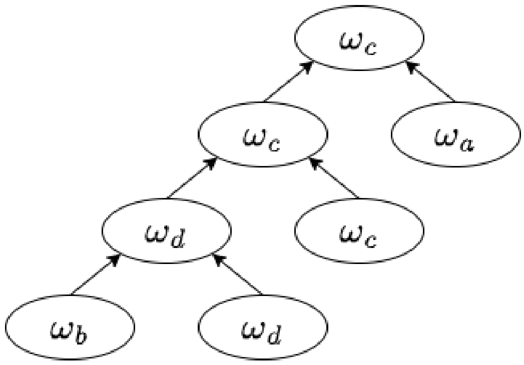
## (2)

存在导致结果为 $\omega_a$ 的议程:




## (3)

存在导致结果为 $\omega_c$ 的议程:




## (4)

康多塞赢家: 对任意议程, 该候选者都是最终赢家.

这个线性序列成对选举中, 不存在康多塞赢家, 因为 (b) 和 (c) 中的两个议程的赢家分别为 $\omega_a$ 和 $\omega_c$, 则不存在唯一最终赢家.

## (5)

应该将 $\{\omega_a, \omega_c\} \to \omega_c$ 改为 $\{\omega_a, \omega_c\} \to \omega_a$.

这样一来 $\omega_a$ 无论在何时对上任何一个候选人, 都能获胜, 因此一定是唯一的最终赢家, 即康多塞赢家.

# 课后作业 4-6

$v(\{a\}) = 0$

$v(\{c\}) = 4$

$v(\{a,b\}) = 6 + 3 + 2 = 11$

$v(\{b,c\}) = 3 + 4 = 7$

$v(\{a,b,c\}) = 6 + 3 + 4 = 13$

# 课后作业 4-7

先计算边际贡献:

$\mu_a(\emptyset) = 12 - 0 = 12$

$\mu_a(\{b\}) = 60 - 18 = 42$

$\mu_a(\{c\}) = 72 - 6 = 66$

$\mu_a(\{b,c\}) = 120 - 48 = 72$

$\mu_b(\emptyset) = 18 - 0 = 18$

$\mu_b(\{a\}) = 60 - 12 = 48$

$\mu_b(\{c\}) = 48 - 6 = 42$

$\mu_b(\{a,c\}) = 120 - 72 = 48$

$\mu_c(\emptyset) = 6 - 0 = 6$

$\mu_c(\{a\}) = 72 - 12 = 60$

$\mu_c(\{b\}) = 48 - 18 = 30$

$\mu_c(\{a,b\}) = 120 - 60 = 60$

计算夏普利值:

$\text{sh}_a = \{2 \times 12 + 42 + 66 + 2 \times 72\}\{3!\} = 46$

$\text{sh}_b = \{2 \times 18 + 48 + 42 + 2 \times 48\}\{3!\} = 37$

$\text{sh}_c = \{2 \times 6 + 60 + 30 + 2 \times 60\}\{3!\} = 37$

# 课后作业 4-8

## (1)

$v(\{A,B\}) = 2$

$v(\{C\}) = 5$

$v(\{A,B,C\}) = 2+4+5 = 11$

### (2)

收益分配 $<1,4,6>$ 在核心中, 因为任何子联盟都无法得到更高的收益

### (3)

收益分配 $<3,4,4>$ 不在核心中, 因为子联盟 $\{B,C\}$ 可以得到更高的收益分配 $<4,5>$

# 课后作业 4-9

$v_{\beta_1}(\{a\}) = 0$

$v_{\beta_1}(\{a,b\}) = 4$

$v_{\beta_1}(\{a,b,c\}) = 4$

$v_{\beta_1}(\{a,b,c,d\}) = 7$

# 课后作业 4-10

维克里拍卖是第二价格, 秘密出价, 一轮拍卖. 即拍卖只有一轮, 在这一轮中, 买方向卖方提交竞拍商品的出价, 没有后续的竞标轮次, 商品分配给出最高价的 Agent 中标者按出价的最二高出价支付.

设 $v_i$ 是某个商品对 Agent $i$ 的价值, $b_i$ 是 Agent $i$ 的出价, 则 Agent i 的收益为

$$p_i = \begin{cases} v_i - \max_{j\neq i} b_j & \text{if } b_i > \max_{j\neq i} b_j \\ 0 & \text{otherwise} \end{cases}$$

下面证明诚实出价是优势策略:

假设 Agent $i$ 的出价 $b_i > v_i$, 即过高出价:

- 如果 $\max_{j\neq i} b_j < v_i$, 那么无论是否诚实出价, 都会中标, 因此诚实出价的竞标策略和过高出价的策略获得同等收益
- 如果 $\max_{j\neq i} b_j = v_i$, 那么两种策略的收益相同, 即诚实出价不中标, 收益为 0; 过高出价中标, 收益为 0
- 如果 $\max_{j\neq i} b_j \geq b_i$, 那么无论是否诚实出价, 都中不了标, 因此两种策略收益相等
- 如果 $v_i < \max_{j\neq b_j} b_j < b_i$, 过高出价会中标, 但是收益是负的, 而诚实策略收益为 0

因此过高出价不如诚实出价.

假设 Agent $i$ 的出价 $b_i < v_i$, 即过低出价:

- 如果 $\max_{j\neq i} b_j \geq v_i$, 那么无论是否诚实出价, 都中不了标, 因此两种策略收益相等
- 如果 $\max_{j\neq i} b_j < b_i$, 那么无论是否诚实出价, 都会中标, 因此诚实出价的竞标策略和过高出价的策略获得同等收益

- 如果 $b_i < \max_{j \neq b_j} b_j < v_i$, 那么诚实出价将会中标, 收益是正的, 而出价过低的收益为 0

因此过低出价不如诚实出价.

# 课后作业 4-11

VCG 机制是激励相容的, 说出真实价值就是优势策略.

我们可以定义一些术语和符号:

- 无差异的价值函数 $v^0$: 对于所有 $Z \subseteq \mathcal{Z}$, 都有 $v^0(Z) = 0$
- 没有 Agent $i$ 的社会福利函数 $sw_{\{-i\}}(Z_1, ..., Z_n, v_1, ..., v_n) = \sum_{j \in \ Ag: j \neq i} v_j(Z_j)$

让每个 Agent 同时宣布一个价值函数 $\hat{v}_i$, VCG 通过如下公式计算最优分配 $Z_1^*, ..., Z_n^*$:

$$Z_1^*, ..., Z_n^* = \underset{Z_1, ..., Z_n \in \text{alloc}(Z, Ag)}{\arg\max} sw(Z_1, ..., Z_n, \hat{v}_1, ..., \hat{v}_n)$$

然后让每个 Agent 支付 $p_i$:

$$\begin{aligned} p_i = {} & sw_{-i}({Z_1}', ..., {Z_n}', \hat{v}_1, ..., v^0, ..., \hat{v}_n) \\ & - sw_{-i}(Z_1^*, ..., Z_n^*, \hat{v}_1, ..., \hat{v}_i, ..., \hat{v}_n) \end{aligned}$$

其中

$${Z_1}', ..., {Z_n}' = \underset{Z_1, ..., Z_n \in \text{alloc}(Z, Ag)}{\arg\max} sw(Z_1, ..., Z_n, \hat{v}_1, ..., v^0, ..., \hat{v}_n)$$

$p_i$ 表示对其他 Agent 因 Agent $i$ 赢奖励配而失去效用的补偿

- ${Z_1}', ..., {Z_n}'$ 是没有 Agent $i$ 参与时的分配结果
- $Z_1^*, ..., Z_n^*$ 是有 Agent $i$ 参与时的分配结果

当 $\mathcal{Z}$ 只包含单个商品时, VCG 机制退化为维克里拍卖.

通过 VCG 机制, 每个 Agent 支付因它们的参与而产生的费用, 没有 Agent 可以通过说谎获利, 因此, VCG 机制为每个 Agent 提供了确保最大化社会福利的优势策略, 即诚实出价.

# 课后作业 4-12

如果协商轮数不确定, 且两个 Agent 都是有耐心的玩家:

Agent 1 在第 0 轮应该提议 $(1, 0)$, 并在之后也一直这样提议, 并且否决 Agent 2 的任何提议.

如此一来, Agent 2 如果一直否决, 就会导致冲突交易; 否则就应该在第 0 轮就同意 Agent 1 的提议.

如果协商轮数不确定, 且两个 Agent 都是耐心有限的玩家, 并且每个 Agent $i$ 有一个折扣因子 $\delta_i, i \in \{1,2\}, 0 \leq \delta_i < 1$, 在第 $t$ 轮, 若 Agent $i$ 得到份额为 $x$, 则其价值为 $\delta_i^t x$:

Agent 1 在第 0 轮应该提议 $\left(\frac{1-\delta_2}{1-\delta_1\delta_2}, \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}\right)$, 这样的话可以在第 0 轮就达成协商.

因为只要 Agent 1 一直提议 $(x, 1-x)$ 并且否决 Agent 2 的任意比它差的提议, Agent 2 的回应方式有如下:

如果第 0 轮否决, 最好情况是 Agent 在第 1 轮的提议在第 2 轮被 Agent 1 同意, Agent 2 的提议不能到达 $1-\delta_1 x$, 否则 Agent 1 不会同意, 这样就意味着如果 Agent 2 在第 0 轮就能得到 $\delta_2(1-\delta_1 x)$, 则就应该在第 0 轮就同意.

所以我们只需 Agent 1 在第 0 轮令 $x = 1-\delta_2(1-\delta_1 x)$ 即可解得 $x = \dfrac{1-\delta_2}{1-\delta_1\delta_2}$

# 课后作业 4-13

**轮流出价协议是一个一对一的协商协议:**

Agent 1 和 Agent 2 进行多轮协商,

- 在第 0 轮, Agent 1 出价 $x^0$
- Agent 2 要么同意, 要么否决. 如果同意, 交易便达成; 如果否决, 进入第 1 轮, Agent 2 出价
- 依次类推, 不断进行

轮流出价协议并不保证最终会达成一致, 如果没有达成一致, 则会产生冲突交易. 并且轮流出价协议有着两个最基本假设, 最差的结果是无法达成一致, 以及每个 Agent 的目标是最大化自己的效用.

**单调让步协议的协商进行多轮:**

在第 $u$ 轮的协商中:

- 两个 Agent 分别从协商集合中提出一项提议
- 如果 Agent 发现另一个 Agent 提出的交易弱优势于他提出的交易, 则达成一致
- 如果没有达成一致, 那么协商进入到下一轮

在第 $u+1$ 轮的协商中:

- Agent 不能提出比第 u 轮的提议对另一个 Agent 更差的提议
- 如果没有 Agent 作出让步, 则协商以交易冲突结束

使用单调让步协议, 在有限轮的协商之后, 可以保证协商结束

- 最后一轮中, 两个 Agent 达成一致或互不让步

- 同时不能保证快速达成一致, 可能的交易数量为 $O\left(2^{|T|}\right)$, 协商可能会进行的轮数与分配的任务数量呈指数关系

# 课后作业 4-14

## (1)

Agent 的第一个提议应该是 Agent 最偏好的交易.

## (2)

在给定的一轮协商中, 度量 Agent 冒冲突风险的意愿, 应该是最不愿意冒冲突风险的 Agent 进行让步.

Agent $i$ 在第 $t$ 轮冒冲突风险的意愿:

$$\text{risk}_i^t = \frac{\text{由于让步并接受 j 的提议导致 i 的效用损失}}{\text{由于没有让步并导致冲突致使 i 的效用损失}}$$

形式地, 有

$$\text{risk}_i^t = \begin{cases} 1 & \text{if utility}_i\left(\delta_i^t\right) = 0 \\ \frac{\text{utility}_i\left(\delta_i^t\right) - \text{utility}_i\left(\delta_j^t\right)}{\text{utility}_i\left(\delta_i^t\right)} & \text{otherwise} \end{cases}$$

这个值在 0~1 之间, 值越大, 表示 Agent $i$ 由冲突遭受的损失越小, 因此更愿意冒冲突风险, 反之亦然.

## (3)

如果一个 Agent 让步, 他应该做出最小的必要的让步, 来改变风险的平衡.

如果遇到风险相同的情况, 可以通过一个 Agent “投掷硬币” 决定谁应该让步.

# 课后作业 4-15

- **无冲突的立场:** $\emptyset$, $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{A, D\}$, $\{A, E\}$, $\{B, C\}$
- **互相辩护的立场:** $\emptyset$, $\{B, C\}$, $\{C, D, E\}$, $\{A, B, C, E\}$, $\{A, C, D, E\}$, $\{B, C, D, E\}$, $\{A, B, C, D, E\}$
- **可采纳的立场:** $\emptyset$, $\{B, C\}$
- **偏好拓展:** $\{B, C\}$
- **轻信接受的论证集合:** $\{B, C\}$
- **怀疑接受的论证集合:** $\{B, C\}$
- **理性拓展:** $\emptyset$

# 课后作业 4-16

使用代码 `p16.py` 计算得到:

- **可采纳的立场:** $\emptyset$, $\{d\}$, $\{e\}$, $\{b,d\}$, $\{c,e\}$, $\{d,f\}$, $\{d,h\}$, $\{e,h\}$, $\{b,d,f\}$, $\{b,d,h\}$, $\{c,e,h\}$, $\{d,f,h\}$, $\{b,d,f,h\}$
- **偏好拓展:** $\{b,d,f,h\}$
- **轻信接受的论证集合:** $\{b,d,f,h\}$
- **怀疑接受的论证集合:** $\{b,d,f,h\}$
- **理性拓展:** $\emptyset$

# 课后作业 4-17

## (1)

联合状态空间 $S=\{\mathrm{Tiger}_{\mathrm{Left}},\mathrm{Tiger}_{\mathrm{Right}}\}$

没有个体状态空间.

## (2)

记 $(\alpha_1,\alpha_2)$ 分别为 Agent 1 和 Agent 2 的动作.

联合动作空间 $A=\{(\mathrm{Open}_{\mathrm{Left}},\mathrm{Open}_{\mathrm{Left}})$, $(\mathrm{Open}_{\mathrm{Left}},\mathrm{Open}_{\mathrm{Right}})$, $(\mathrm{Open}_{\mathrm{Left}},\mathrm{Listen})$, $(\mathrm{Open}_{\mathrm{Right}},\mathrm{Open}_{\mathrm{Left}})$, $(\mathrm{Open}_{\mathrm{Right}},\mathrm{Open}_{\mathrm{Right}})$, $(\mathrm{Open}_{\mathrm{Right}},\mathrm{Listen})$, $(\mathrm{Listen},\mathrm{Open}_{\mathrm{Left}})$, $(\mathrm{Listen},\mathrm{Open}_{\mathrm{Right}})$, $(\mathrm{Listen},\mathrm{Listen})\}$

Agent $i$ 的个体动作空间 $A^i=\{\mathrm{Open}_{\mathrm{Left}},\mathrm{Open}_{\mathrm{Right}},\mathrm{Listen}\}$

## (3)

联合观察空间 $O=\{(\mathrm{Roar}_{\mathrm{Left}},\mathrm{Roar}_{\mathrm{Left}}),(\mathrm{Roar}_{\mathrm{Left}},\mathrm{Roar}_{\mathrm{Right}}),(\mathrm{Roar}_{\mathrm{Right}},\mathrm{Roar}_{\mathrm{Left}}),(\mathrm{Roar}_{\mathrm{Right}},\mathrm{Roar}_{\mathrm{Right}})\}$

Agent $i$ 的个体观察空间 $O^i=\{\mathrm{Roar}_{\mathrm{Left}},\mathrm{Roar}_{\mathrm{Right}}\}$

# 课后作业 4-18

这一问中，借助 OpenMarkov 创建了一个如 图 1 用于描述 Dec-Tiger 问题的 Dec-POMDP，通过分别构建了 State、Action、Observation 与 Utility 完成了这项任务。

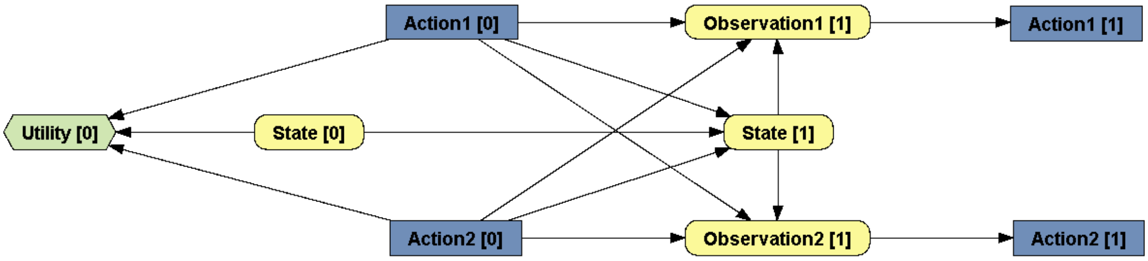



图 1: State、Action、Observation 与 Utility 之间的连接关系


图 2: State 对应的状态有 Tiger_Left 与 Tiger_Right

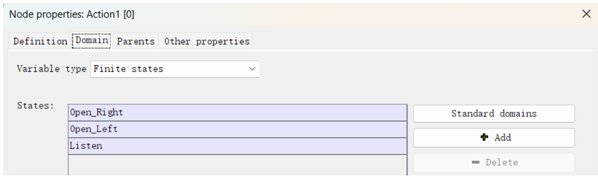



图 3: Action 可以执行的动作有 Open_Left、Open_Right 与 Listen

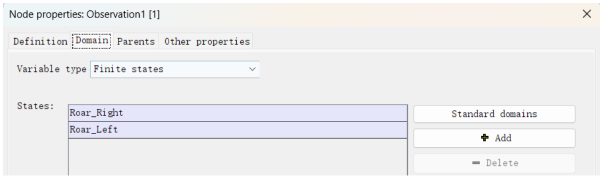
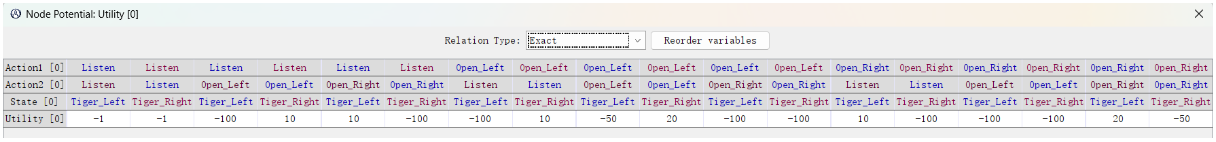



图 4: Observation 可以观察到 Roar_Left 与 Roar_Right




| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Action1 [0] | Listen | Listen | Listen | Listen | Listen | Listen | Open_Left | Open_Left | Open_Left | Open_Left | Open_Left | Open_Left | Open_Right | Open_Right | Open_Right | Open_Right | Open_Right | Open_Right |
| Action2 [0] | Listen | Listen | Open_Left | Open_Left | Open_Right | Open_Right | Listen | Listen | Open_Left | Open_Left | Open_Right | Open_Right | Listen | Listen | Open_Left | Open_Left | Open_Right | Open_Right |
| State [0] | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right | Tiger_Left | Tiger_Right |
| Utility [0] | -1 | -1 | -100 | 10 | 10 | -100 | -100 | 10 | -50 | 20 | -100 | -100 | 10 | -100 | -100 | -100 | 20 | -50 |

图 5: Utility 效用通过当前 State 与 Action 计算得到的奖励

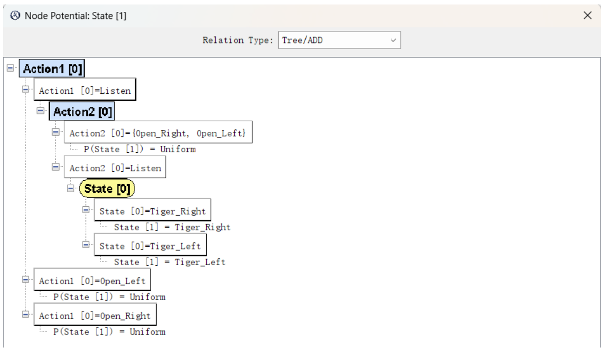



图 6: t + 1 时刻 State 由 t 时刻的 State 与 Action 得到

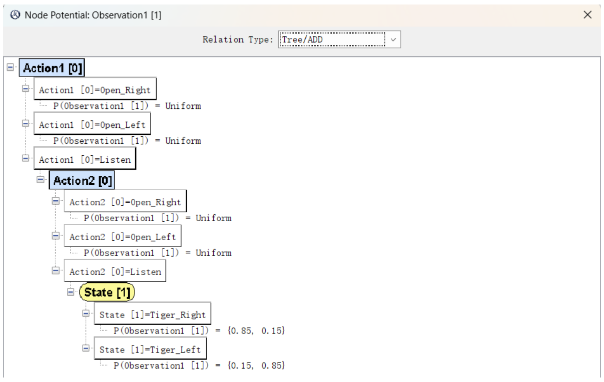



图 7: t + 1 时刻 Observation 由 t + 1 时刻的 State 与 t 时刻的 Action 得到

如 图 2、 图 3、 图 4 所示，我们分别给 State、Action 与 Observation 加上了各自的离散取值。

随后即如 图 5 中为 Utility 加入对应的奖励。按照我所理解的题意，在有一个 Agent 打开门的时候，即使另一个 Agent 在 Listen，也不会产生 Listen 的开销，即有一方在 Listen

而另一方打开门发现宝箱的情况总奖励为 10，有一方在 Listen 而另一方打开门碰到老虎的情况总奖励为 -100，且后者奖励与两个 Agent 同时打开两扇门奖励同为 -100。只有在两个 Agent 均在 Listen 的时候才会有 -1 的开销。

较为复杂的就是 图 6 与 图 7 的策略树。 图 6 表示只有在两个 Agent 的 Action 均为 Listen 的时候，t + 1 时刻的 State 才会维持与 t 时刻的 State 相同，其他情况下以均匀随机得到新 State； 图 7 表示只有在两个 Agent 的 Action 均为 Listen 的时候，t + 1 时刻的 Observation 才会根据 t + 1 时刻的 State 以 0.85 的概率给出正确的 Observation。

其次我们注意到，直接保存后得到的文件内容的一部分为：

```
<Potential type="Table" role="conditionalProbability">
  <UtilityVariable name="Utility" timeSlice="0" />
  <Variables>
    <Variable name="Action1" timeSlice="0" />
    <Variable name="Action2" timeSlice="0" />
    <Variable name="State" timeSlice="0" />
  </Variables>
  <Values>-1.0 -100.0 10.0 -100.0 -50.0 -100.0 10.0 -100.0 20.0 -1.0 10.0
-100.0 10.0 20.0 -100.0 -100.0 -100.0 -50.0</Values>
</Potential>
```

但是这样并不能在之后的小间中执行，因此我们需要将 `conditionalProbability` 改为 `utility` 才能正确执行。

## 课后作业 4-19

### (1)

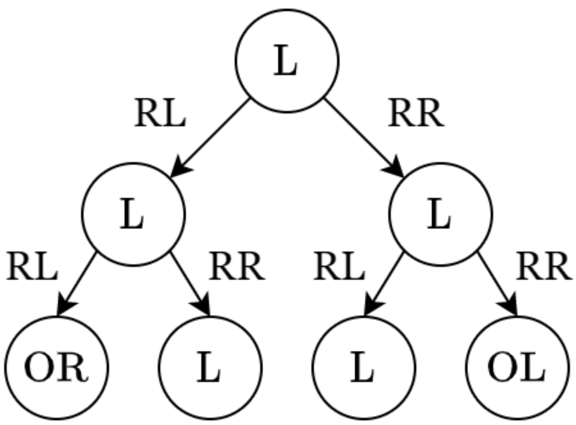



图 8: 对于 Dec-Tiger 问题以单个 Agent 的视角得到的一种 horizon h=3 策略树

## (2)

以下推导均基于 **两个 Agent 有着相同的决策树时奖励最高** 这一个由题意易得的事实进行的推导。

当 `h=1` 时，最优策略是:

- **Agent 1**: $() \to \text{Listen}.$
- **Agent 2**: $() \to \text{Listen}.$

即两个 Agent 总是 Listen。

可以算出这种策略的期望奖励为：$-1$。

优于两个 Agent 均选择 $\text{Open}_{\text{Left}}$ 或 $\text{Open}_{\text{Right}}$ 时 $0.5 \times (-50) + 0.5 \times 20 = -15$ 的奖励。

当 `h=2` 时，最优策略是:

- **Agent 1**: $() \to \text{Listen}, (\text{Roar}_{\text{Left}}) \to \text{Listen}, (\text{Roar}_{\text{Right}}) \to \text{Listen}.$
- **Agent 2**: $() \to \text{Listen}, (\text{Roar}_{\text{Left}}) \to \text{Listen}, (\text{Roar}_{\text{Right}}) \to \text{Listen}.$

可以算出这种策略的期望奖励为

$$\begin{aligned}
&0.5 \times \left(-1 + 0.85^2 \times (-1) + 2 \times 0.85 \times 0.15 \times (-1) + 0.15^2 \times (-1)\right)\\
&+ 0.5 \times \left(-1 + 0.15^2 \times (-1) + 2 \times 0.15 \times 0.85 \times (-1) + 0.85^2 \times (-1)\right)\\
&= -2
\end{aligned}$$

优于 $() \to \text{Listen}, (\text{Roar}_{\text{Left}}) \to \text{Open}_{\text{Right}}, (\text{Roar}_{\text{Right}}) \to \text{Open}_{\text{Left}}$ 的期望奖励

$$\begin{aligned}
&0.5 \times \left(-1 + 0.85^2 \times 20 + 2 \times 0.85 \times 0.15 \times (-100) + 0.15^2 \times (-50)\right)\\
&+ 0.5 \times \left(-1 + 0.15^2 \times (-50) + 2 \times 0.15 \times 0.85 \times (-100) + 0.85^2 \times 20\right)\\
&= -13.175
\end{aligned}$$

同样优于 $() \to \text{Listen}, (\text{Roar}_{\text{Left}}) \to \text{Open}_{\text{Right}}, (\text{Roar}_{\text{Right}}) \to \text{Listen}$ 的期望奖励

$$\begin{aligned}
&0.5 \times \left(-1 + 0.85^2 \times 20 + 2 \times 0.85 \times 0.15 \times 10 + 0.15^2 \times (-1)\right)\\
&+ 0.5 \times \left(-1 + 0.15^2 \times (-50) + 2 \times 0.15 \times 0.85 \times (-100) + 0.85^2 \times (-1)\right)\\
&= -6.185
\end{aligned}$$

其他情况同理。

当 `h=3` 时，最优策略是两个 Agent 均使用策略树:

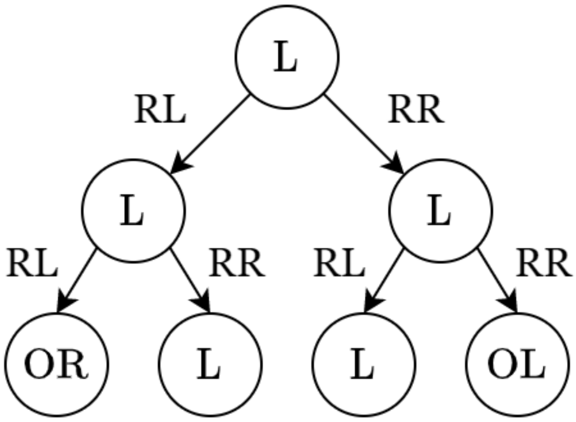



图 9: h = 3 时的最优策略树

可以算出这种策略的期望奖励为

$$
\begin{aligned}
&-1-1+a^2c^2\times 20+2a^2cd\times 10+a^2d^2\times(-100)+2abc^2\times 10\\
&+4abcd\times(-1)+2abd^2\times(-100)+b^2c^2\times(-100)+2b^2cd\times(-100)+b^2d^2\times(-50)\\
&=7.6357875
\end{aligned}
$$

其中 $a=0.85, b=0.15, c=0.85, d=0.15$。

由 `h = 1` 与 `h = 2` 时的最优策略可知一二层均应为 Listen，而第三层又易知最优策略即为如 图 9 所示的策略树，因此可知最优策略即为期望奖励为 7.6357875 的如 图 9 所示策略树。

## (3)

由联合策略的个数公式

$$O\left(m\times\left|\mathcal{A}^{\max}\right|^{\frac{\left|\mathcal{O}^{\max}\right|^{T}-1}{\left|\mathcal{O}^{\max}\right|-1}}\right)$$

带入 $m=2, \left|\mathcal{A}^{\max}\right|=3, \left|\mathcal{O}^{\max}\right|=2, T=3$ 可得

$$m\times\left|\mathcal{A}^{\max}\right|^{\frac{\left|\mathcal{O}^{\max}\right|^{T}-1}{\left|\mathcal{O}^{\max}\right|-1}}=2\times 3^{\frac{2^3-1}{2-1}}=4374$$

因此联合策略空间大小为 4374。

# 课后作业 4-20

## (1)

使用默认值 (GMAA-MAAstar using a BGIP-BFS solver and a INVALIDQHEUR heuristic) 计算的结果为:

**当 `h = 1` 时的结果:**

```
#horvalue     Value simul.wc-tot.ut-tot.st-tot.wc-GMAAut-GMAAst-GMAAwc-Qcom
ut-Qcomst-Qcomwc-initut-initst-initnrEvalBGjpolsmaxPoolSizeJPindexk - times
are in ticks (1/100s)and are GMAAF times (do not include heuristic
computation)
1-1.000000-1.000000110000000011000PartialJPPV, past R=-1, 02147483647
# h 1 avg GMAA time (s): 0.000000 avg value: -1.000000
```

```
JointPolicyPureVector:
JPolComponent_VectorImplementation index 0
Policy for agent 0 (index 0):
() --> Listen
Policy for agent 1 (index 0):
() --> Listen

digraph policyAgent0 {
edge [dir=none];
node0 [ label="Listen" ];
}

digraph policyAgent1 {
edge [dir=none];
node0 [ label="Listen" ];
}

Sampled value =  -1 (nrSimRuns=10000)
Computed value = -1
```

**当 `h = 2` 时的结果:**

```
#horvalue     Value simul.wc-tot.ut-tot.st-tot.wc-GMAAut-GMAAst-GMAAwc-Qcom
ut-Qcomst-Qcomwc-initut-initst-initnrEvalBGjpolsmaxPoolSizeJPindexk - times
are in ticks (1/100s)and are GMAAF times (do not include heuristic
computation)
2-2.000000-2.000000110000000011009PartialJPPV, past R=-2, 02147483647
# h 2 avg GMAA time (s): 0.000000 avg value: -2.000000
```

```
JointPolicyPureVector:
JPolComponent_VectorImplementation index 0
Policy for agent 0 (index 0):
() --> Listen
```

```
(Roar_Left) --> Listen
(Roar_Right) --> Listen
Policy for agent 1 (index 0):
() --> Listen
(Roar_Left) --> Listen
(Roar_Right) --> Listen

digraph policyAgent0 {
edge [dir=none];
node0 [ label="Listen" ];
node1 [ label="Listen" ];
node2 [ label="Listen" ];
node0 -> node1 [label="Roar_Left"];
node0 -> node2 [label="Roar_Right"];
}

digraph policyAgent1 {
edge [dir=none];
node0 [ label="Listen" ];
node1 [ label="Listen" ];
node2 [ label="Listen" ];
node0 -> node1 [label="Roar_Left"];
node0 -> node2 [label="Roar_Right"];
}

Sampled value =  -2 (nrSimRuns=10000)
Computed value = -2
```

**当 `h = 3` 时的结果：**

```
#horvalue     Value simul.wc-tot.ut-tot.st-tot.wc-GMAAut-GMAAst-GMAAwc-Qcom
ut-Qcomst-Qcomwc-initut-initst-initnrEvalBGjpolsmaxPoolSizeJPindexk - times
are in ticks (1/100s)and are GMAAF times (do not include heuristic
computation)
37.6357877.623900970870000100012PartialJPPV, past R=7.63579, 63452
2147483647
# h 3 avg GMAA time (s): 0.070000 avg value: 7.635787
```

```
JointPolicyPureVector:
JPolComponent_VectorImplementation index 63452
Policy for agent 0 (index 29):
() --> Listen
(Roar_Left) --> Listen
(Roar_Right) --> Listen
(Roar_Left,Roar_Left) --> Open_Left
(Roar_Left,Roar_Right) --> Listen
(Roar_Right,Roar_Left) --> Listen
(Roar_Right,Roar_Right) --> Open_Right
Policy for agent 1 (index 29):
() --> Listen
```

```
(Roar_Left) --> Listen
(Roar_Right) --> Listen
(Roar_Left,Roar_Left) --> Open_Left
(Roar_Left,Roar_Right) --> Listen
(Roar_Right,Roar_Left) --> Listen
(Roar_Right,Roar_Right) --> Open_Right

digraph policyAgent0 {
edge [dir=none];
node0 [ label="Listen" ];
node1 [ label="Listen" ];
node2 [ label="Listen" ];
node3 [ label="Open_Left" ];
node4 [ label="Listen" ];
node5 [ label="Listen" ];
node6 [ label="Open_Right" ];
node0 -> node1 [label="Roar_Left"];
node0 -> node2 [label="Roar_Right"];
node1 -> node3 [label="Roar_Left"];
node1 -> node4 [label="Roar_Right"];
node2 -> node5 [label="Roar_Left"];
node2 -> node6 [label="Roar_Right"];
}

digraph policyAgent1 {
edge [dir=none];
node0 [ label="Listen" ];
node1 [ label="Listen" ];
node2 [ label="Listen" ];
node3 [ label="Open_Left" ];
node4 [ label="Listen" ];
node5 [ label="Listen" ];
node6 [ label="Open_Right" ];
node0 -> node1 [label="Roar_Left"];
node0 -> node2 [label="Roar_Right"];
node1 -> node3 [label="Roar_Left"];
node1 -> node4 [label="Roar_Right"];
node2 -> node5 [label="Roar_Left"];
node2 -> node6 [label="Roar_Right"];
}

Sampled value =  7.6239 (nrSimRuns=10000)
Computed value = 7.63579
```

**当 `h = 4` 时的结果：**

```
#horvalue     Value simul.wc-tot.ut-tot.st-tot.wc-GMAAut-GMAAst-GMAAwc-Qcom
ut-Qcomst-Qcomwc-initut-initst-initnrEvalBGjpolsmaxPoolSizeJPindexk - times
are in ticks (1/100s)and are GMAAF times (do not include heuristic
computation)
48.4214498.36760057930645792834122579306257928331220002100 46PartialJPPV,
```

```
past R=8.42145, 3140975961221474836 47
# h 4 avg GMAA time (s): 57928.330000 avg value: 8.421449
```

```
JointPolicyPureVector:
JPolComponent_VectorImplementation index 31409759612
Policy for agent 0 (index 2189):
() --> Listen
(Roar_Left) --> Listen
(Roar_Right) --> Listen
(Roar_Left,Roar_Left) --> Listen
(Roar_Left,Roar_Right) --> Listen
(Roar_Right,Roar_Left) --> Listen
(Roar_Right,Roar_Right) --> Listen
(Roar_Left,Roar_Left,Roar_Left) --> Open_Left
(Roar_Left,Roar_Left,Roar_Right) --> Listen
(Roar_Left,Roar_Right,Roar_Left) --> Listen
(Roar_Left,Roar_Right,Roar_Right) --> Listen
(Roar_Right,Roar_Left,Roar_Left) --> Listen
(Roar_Right,Roar_Left,Roar_Right) --> Listen
(Roar_Right,Roar_Right,Roar_Left) --> Listen
(Roar_Right,Roar_Right,Roar_Right) --> Open_Right
Policy for agent 1 (index 2189):
() --> Listen
(Roar_Left) --> Listen
(Roar_Right) --> Listen
(Roar_Left,Roar_Left) --> Listen
(Roar_Left,Roar_Right) --> Listen
(Roar_Right,Roar_Left) --> Listen
(Roar_Right,Roar_Right) --> Listen
(Roar_Left,Roar_Left,Roar_Left) --> Open_Left
(Roar_Left,Roar_Left,Roar_Right) --> Listen
(Roar_Left,Roar_Right,Roar_Left) --> Listen
(Roar_Left,Roar_Right,Roar_Right) --> Listen
(Roar_Right,Roar_Left,Roar_Left) --> Listen
(Roar_Right,Roar_Left,Roar_Right) --> Listen
(Roar_Right,Roar_Right,Roar_Left) --> Listen
(Roar_Right,Roar_Right,Roar_Right) --> Open_Right

digraph policyAgent0 {
edge [dir=none];
node0 [ label="Listen" ];
node1 [ label="Listen" ];
node2 [ label="Listen" ];
node3 [ label="Listen" ];
node4 [ label="Listen" ];
node5 [ label="Listen" ];
node6 [ label="Listen" ];
node7 [ label="Open_Left" ];
node8 [ label="Listen" ];
node9 [ label="Listen" ];
node10 [ label="Listen" ];
node11 [ label="Listen" ];
```

```
node12 [ label="Listen" ];
node13 [ label="Listen" ];
node14 [ label="Open_Right" ];
node0 -> node1 [label="Roar_Left"];
node0 -> node2 [label="Roar_Right"];
node1 -> node3 [label="Roar_Left"];
node1 -> node4 [label="Roar_Right"];
node2 -> node5 [label="Roar_Left"];
node2 -> node6 [label="Roar_Right"];
node3 -> node7 [label="Roar_Left"];
node3 -> node8 [label="Roar_Right"];
node4 -> node9 [label="Roar_Left"];
node4 -> node10 [label="Roar_Right"];
node5 -> node11 [label="Roar_Left"];
node5 -> node12 [label="Roar_Right"];
node6 -> node13 [label="Roar_Left"];
node6 -> node14 [label="Roar_Right"];
}

digraph policyAgent1 {
edge [dir=none];
node0 [ label="Listen" ];
node1 [ label="Listen" ];
node2 [ label="Listen" ];
node3 [ label="Listen" ];
node4 [ label="Listen" ];
node5 [ label="Listen" ];
node6 [ label="Listen" ];
node7 [ label="Open_Left" ];
node8 [ label="Listen" ];
node9 [ label="Listen" ];
node10 [ label="Listen" ];
node11 [ label="Listen" ];
node12 [ label="Listen" ];
node13 [ label="Listen" ];
node14 [ label="Open_Right" ];
node0 -> node1 [label="Roar_Left"];
node0 -> node2 [label="Roar_Right"];
node1 -> node3 [label="Roar_Left"];
node1 -> node4 [label="Roar_Right"];
node2 -> node5 [label="Roar_Left"];
node2 -> node6 [label="Roar_Right"];
node3 -> node7 [label="Roar_Left"];
node3 -> node8 [label="Roar_Right"];
node4 -> node9 [label="Roar_Left"];
node4 -> node10 [label="Roar_Right"];
node5 -> node11 [label="Roar_Left"];
node5 -> node12 [label="Roar_Right"];
node6 -> node13 [label="Roar_Left"];
node6 -> node14 [label="Roar_Right"];
}
```

```
Sampled value =  8.3676 (nrSimRuns=10000)
Computed value = 8.42145
```

可以看出计算得到的结果与我们的推测一致，例如 `h = 1` 时最优值为 $-1$，`h = 2` 时最优值为 $-2$，以及 `h = 3` 时最优值为 7.63579。

并且在 `h = 1` 到 `h = 3` 时均耗时不超过 0.1 秒就能跑出最后结果，但是 `h = 4` 的时候跑了 57928 秒，也即 16 个小时，从这里我们也可以看出双指数增长的威力。

## (2)

我们仍然使用 GMAA 来求解，其中主要参数有：

1. `-B`：贝叶斯博弈的求解器，主要包括：
   - `BFS`：宽度优先暴力搜索；
   - `AM`：交替最大化的近似解；
   - `CE`：交叉熵优化的近似解；
   - `MP`：Max-Plus 的近似解；
   - `BnB`：Branch-and-Bound 求解；
   - `Random`：随机，用于测试。
2. `-G`：该方法是否执行完整的回溯启发式搜索，是否仅对从根到叶的一条路径进行采样，或者是否在两者之间执行某些操作，主要包括：
   - `MAAstar`：完全回溯 (MAA*) 搜索的选项，执行搜索节点的增量展开；
   - `FSPC`：选择前向扫描策略，在搜索树中的每个节点上，只扩展最有希望的子节点；
   - `kGMAA`：这使用参数 k (使用选项 -k 指定) 在每个节点上仅扩展 k 个最有希望的子节点；
   - `MAAstarClassic`：使用内置 BFS 求解器的旧版本 (并且不进行增量扩展)。
3. `-Q`：确定启发式函数，其中包括 `QMDP`、`QPOMDP`、`QBG`、`QMDPc`、`QPOMDPav`、`QBGav`、`QHybrid`、`QPOMDPhybrid`、`QBGhybrid`、`QBGTreeIncPrune`、`QBGTreeIncPruneBnB`。

在前一问中，不设定使用默认参数执行即为使用参数 `-B BFS -G MAAstar -Q` `QMDP`，其中 BFS 为暴力搜索，MAAstar 为完全回溯搜索，QMDP 是在不给定启发式函数时的使用的默认函数。

为了比较不同 horizon 和不同解法的结果，使用代码 `p20.py` 计算得到：

当 `horizon = 4` 时，运行结果如下，其中是否最优指生成的策略树是否与默认 BFS 搜索得到的结果一致。

| 参数 | | | 结果 | | |
|---|---|---|---|---|---|
| `-B` | `-G` | `-Q` | 是否最优 | 计算值 | 耗时 |
| BFS | MAAstar | QMDP | 是 | 8.421449 | 57928.33 s |
| AM | FSPC | QMDP | 否 | 6.635787 | 0.01 s |
| AM | FSPC | QPOMDP | 否 | 7.104371 | 0.00 s |
| AM | kGMAA | QMDP | 否 | 6.635787 | 0.01 s |
| AM | kGMAA | QPOMDP | 是 | 8.421449 | 0.00 s |
| BnB | FSPC | QMDP | 否 | 6.635787 | 0.01 s |
| BnB | FSPC | QPOMDP | 是 | 8.421449 | 0.13 s |
| BnB | MAAstar | QMDP | 是 | 8.421449 | 3.54 s |
| BnB | MAAstar | QPOMDP | 是 | 8.421449 | 0.20 s |
| BnB | kGMAA | QMDP | 否 | 6.635787 | 0.01 s |
| BnB | kGMAA | QPOMDP | 是 | 8.421449 | 0.12 s |
| CE | FSPC | QMDP | 否 | 6.635787 | 0.20 s |
| CE | FSPC | QPOMDP | 是 | 8.421449 | 0.19 s |
| CE | kGMAA | QMDP | 否 | 6.635787 | 0.20 s |
| CE | kGMAA | QPOMDP | 是 | 8.421449 | 0.20 s |
| MP | FSPC | QMDP | 否 | 6.635787 | 0.01 s |
| MP | FSPC | QPOMDP | 是 | 8.421449 | 0.00 s |
| MP | kGMAA | QMDP | 否 | 6.635787 | 0.01 s |
| MP | kGMAA | QPOMDP | 是 | 8.421449 | 0.00 s |

图 10: h = 4 时，不同参数的 GMAA 计算结果

就 `-B` 参数来说，`BFS` 暴力搜索是最慢的，仅仅是在 `h = 4` 的情况下就要搜索 16 个小时。而 `AM`、`CE` 与 `MP` 等近似求解器的计算用时很短，但是大部分时候都达不到最优策略。而同样是近似求解器的 `BnB` 表现就好很多，在运算速度和计算结果上做了一个较好的折中。

就 `-G` 参数来说，大部分算法都无法采用 `MAAstar` 搜索，仅有 `BFS` 与 `BnB` 能够使用，并且求解出来的结果也一定是最优的，但是一定要配合上 `BnB` 才能有一个较快的求解速度。而 `FSPC` 与 `kGMAA` 都是只保留一部分最有希望的节点，其中 `FSPC` 最为激进，只扩展最有希望的子节点，因此求解速度快，但是大部分时候都达不到最优；而 `kGMAA` 允许你设定 k 值以保留 k 个最有希望的子节点，因此能在线性时间的计算耗时和相对更好的计算结果中做出平衡。

就 `-Q` 参数来说，`QPOMDP` 在计算结果精确度上总是优于 `QMDP`，但是计算耗时则不一定，部分情况 `QPOMDP` 较快，部分情况 `QMDP` 较快。

当 `horizon = 5` 时，运行结果如下

| 参数 | | | 结果 | |
|---|---|---|---|---|
| `-B` | `-G` | `-Q` | 计算值 | 耗时 |
| AM | FSPC | QMDP | 5.635787 | 0.03 s |
| AM | FSPC | QPOMDP | 5.635787 | 0.01 s |
| AM | kGMAA | QMDP | 5.635787 | 0.04 s |
| AM | kGMAA | QPOMDP | 5.635787 | 0.01 s |
| CE | FSPC | QMDP | 5.635787 | 0.63 s |
| CE | FSPC | QPOMDP | 5.635787 | 0.57 s |
| CE | kGMAA | QMDP | 5.635787 | 0.62 s |
| CE | kGMAA | QPOMDP | 5.635787 | 0.57 s |
| MP | FSPC | QMDP | 5.635787 | 0.04 s |
| MP | FSPC | QPOMDP | 5.635787 | 0.01 s |
| MP | kGMAA | QMDP | 5.635787 | 0.03 s |
| MP | kGMAA | QPOMDP | 5.635787 | 0.01 s |

图 11: $h = 5$ 时，不同参数的 GMAA 计算结果

这时候使用 `BFS`、`BnB` 以及 `MAAstar` 在计算时间开销上都是不可接受的，因此只使用了线性时间内完成的求解方法。

对于 horizon 无穷的情况，设定折扣系数为 0.9，使用 Perseus 求解，命令如下

```
./src/solvers/Perseus ./problems/Dec-Tiger.pgmx --inf --discount=0.9 -n50
```

结果如下：

```
PerseusPOMDP: iteration    155 |V| 3 sumV/nrB 66.7072 V0 65.8027 (best
65.8027)
Added vector for 34 (V 65.8028 improved 46)
Added vector for 3 (V 77.1083 improved 2)
Added vector for 40 (V 77.1083 improved 2)
```