Finding Friend and Foe in Multi-Agent Games

Jack Serrino; Max Kleinman-Weiner; David C.Parkes; Joshua B.Tenenbaum
NIPS 2019 (Harward, MIT)
报告人：汪永毅

1. Avalon 游戏介绍

故事背景：[以「阿瓦隆」，一窺亞瑟王傳奇 – Flere 瞻前顧後 (wordpress.com)]

阵营划分：

玩家共5人，分为2个阵营：正方3人（1 Merlin & 2 Resistance），反方2人 (1 Spy & 1 Assasin)。

角色特点：

各个角色都知道自己的身份。

Merlin知道反方2人，Resistance仅知道自己的身份。

Spy & Assasin互相知道身份。

Assasin可以在游戏结束后指认Merlin，若指认正确则反方取得胜利。

游戏流程：

1. 随机发放5张身份牌M(erlin), R(esistance), R(esistance), S(py), A(ssasin)。
2. 通过睁眼闭眼以及竖大拇指的方式让M知道反方2人，S & A互认。
3. 随机选择一人作为起始队长，指定2-3人组队做任务，可以包含或不包含队长自己。

   (任务执行至多5次，每次队长需指定人数分别是2,3,3,2,3)。
4. 5人同时投票，按照多数决定是否同意队长指定的队伍来执行此次任务。
   1. 如果决定要执行，则队伍中人秘密投票决定任务成功或失败。逆时针下一个人接任队长，开启下一次任务。

   (队伍中正方成员必须投成功，反方成员可投成功或失败。任务成功当且仅当所有票都是成功。仅公布计票结果)。

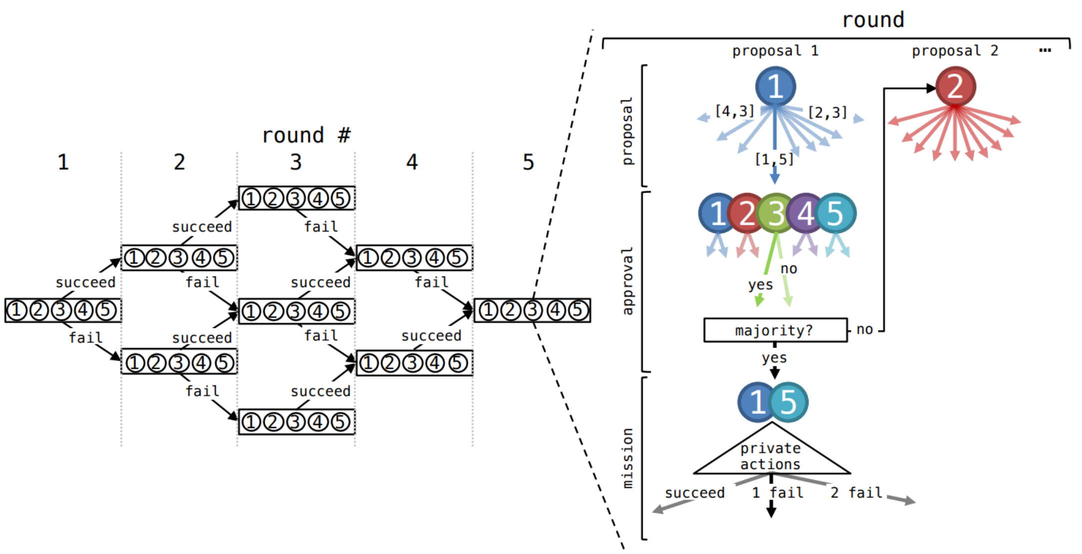
   2. 如果决定不执行，则由逆时针下一个人当队长，来重新指定此次任务的队伍。

   (不执行最多可以连续进行4次，当第5人指定队伍时，不投票强制执行任务)。

胜负判定：

任务成功次数达到3且Merlin未被Assasin认出则正方胜。

任务失败次数达到3或者Assasin成功指认出Merlin则反方胜。



各Round组队人数分别为2,3,3,2,3

Round中轮流当队长提出Proposal，投票决定是否执行任务，执行则进入下一Round

游戏特点:

非完美信息（同时决策）、非完全信息（阵营未知导致支付函数不确定）。

信息集数目10^{56}，大于有限注德扑10^{14}，国际象棋10^{47}。

与传统5人Avalon的区别:

1. 为便于算法运行，不考虑玩家发言阶段。
2. 身份称呼：（先知-梅林，骑士-派西维尔，忠臣-盖亚思，假先知-莫甘娜，刺客-阿德规文）。
3. 传统Avalon正方有骑士-派西维尔，可以看到先知-梅林和假先知-莫甘娜，但无法区分二者。

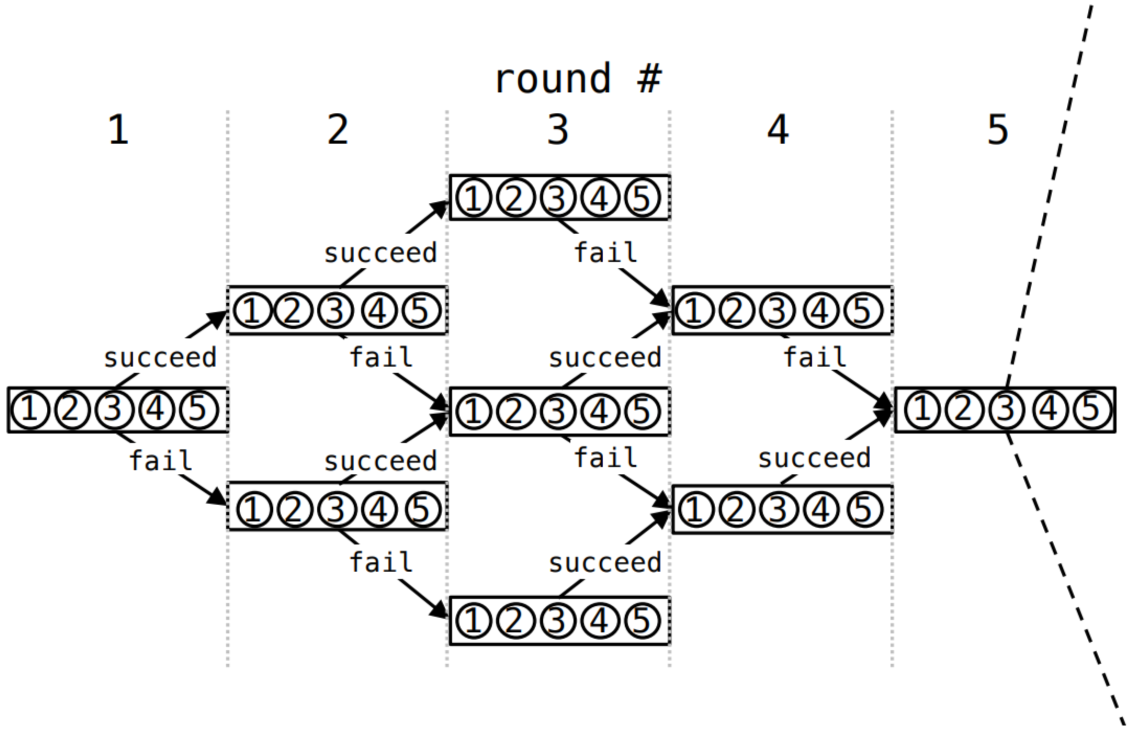
2. 论文概述

论文提出了DeepRole算法，用于求解5人Avalon游戏。

算法思路：

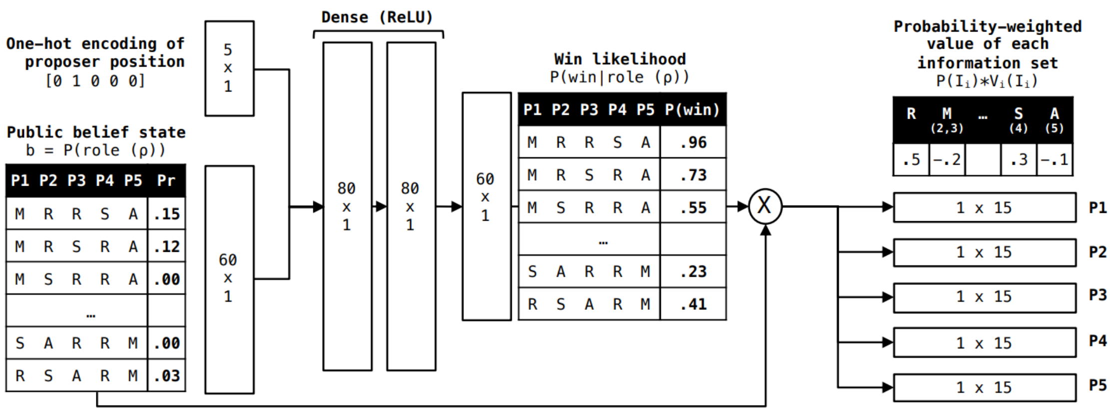
在DeepStack的基础上改进：CFR+规划算法结合逻辑推理、用DNN为信息集估值减小博弈树规模。

将Avalon划分为$9\times 5=45$个子问题（按照任务成功失败数目分为9种情况，每种情况可能经历5个不同的队长提议组队过程）。



9*5=45个子问题

每个子问题对应一个神经网络，接收历史与$\binom{5}{1,2,1,1}=60$种角色分配情况的公共信念分布b，输出各玩家各信息集的价值估计。



60种身份分配，15种信息集

上图中计算角色分配情况的公共信念分布b采用贝叶斯定理：

$$b(\rho|h)=\frac{Pr\{\rho,h|\sigma\}}{Pr\{h|\sigma\}}\propto b(\rho)Pr\{h|\rho,\sigma\}=b(\rho)(1-\mathbb{1}(h\vdash\neg\rho))\prod_{i\in\{1,\dots,p\}}\pi_i^{\sigma}(I_i(h,\rho))$$

网络训练方法：随机生成数据，自后向前逐阶段训练。

3. 算法细节

Algorithm 1 DeepRole depth-limited CFR

1: **INPUT** h (root public game history); $\mathbf{b}$ (root public belief); n (# iterations); d (averaging delay); NN$[h]$ (neural networks that approximate CFVs from h)

Init regrets $\forall I, r_I[a] \leftarrow 0$, Init cumulative strategies $\forall I, s_I[a] \leftarrow 0$

2: **procedure** SOLVESITUATION$(h, \mathbf{b}, n, d)$
3: $\quad \vec{u}_{1...p} \leftarrow \vec{0}$
4: $\quad$ **for** $i = 1$ to n **do**
5: $\quad\quad w_i \leftarrow \max(i - d, 0)$
6: $\quad\quad \vec{u}_{1...p} \leftarrow \vec{u}_{1...p}+$MODIFIEDCFR+$(h, \mathbf{b}, w_i, \vec{1}_{1...p})$
7: $\quad$ **end for**
8: $\quad$ **return** $\vec{u}_{1...p}$ / $\sum w_i$
9: **end procedure**

10: **procedure** MODIFIEDCFR+$(h, \mathbf{b}, w, \vec{\pi}_{1...p})$
11: $\quad$ **if** $h \in Z$ **then**
12: $\quad\quad$ **return** TERMINALCFVS$(h, \mathbf{b}, \vec{\pi}_{1...p})$
13: $\quad$ **end if**
14: $\quad$ **if** $h \in$ NN **then**
15: $\quad\quad$ **return** NEURALCFVS$(h, \mathbf{b}, \vec{\pi}_{1...p})$
16: $\quad$ **end if**
17: $\quad \vec{u}_{1...p} \leftarrow \vec{0}$
18: $\quad$ **for** $i \in P'(h)$ **do** $\triangleright$ A strategy must be calculated for all moving players at public history h
19: $\quad\quad \vec{I}_i \leftarrow \text{lookupInfosets}_i(h)$
20: $\quad\quad \vec{\sigma}_i \leftarrow \text{regretMatching+}(\vec{I}_i)$
21: $\quad$ **end for**
22: $\quad$ **for** public observation $o \in O(h)$ **do**
23: $\quad\quad \vec{a}_{1...p} \leftarrow \text{deduceActions}(h, o)$
24: $\quad\quad$ **for** $i \in P'(h)$ **do**
25: $\quad\quad\quad \vec{\pi}_i \leftarrow \vec{\sigma}_i[\vec{a}_i] \odot \vec{\pi}_i$
26: $\quad\quad$ **end for**
27: $\quad\quad \vec{u}'_{1...p} \leftarrow$ MODIFIEDCFR+$(ho, \mathbf{b}, w, \vec{\pi}_{1...p})$
28: $\quad\quad$ **for** each player i **do**
29: $\quad\quad\quad$ **if** $i \in P'(h)$ **then**
30: $\quad\quad\quad\quad \vec{m}_i[\vec{a}_i] \leftarrow \vec{m}_i[\vec{a}_i] + \vec{u}_i$
31: $\quad\quad\quad\quad \vec{u}_i \leftarrow \vec{u}_i + \vec{\sigma}_i[\vec{a}_i] \odot \vec{u}'_i$
32: $\quad\quad\quad$ **else**
33: $\quad\quad\quad\quad \vec{u}_i \leftarrow \vec{u}_i + \vec{u}'_i$
34: $\quad\quad\quad$ **end if**
35: $\quad\quad$ **end for**
36: $\quad$ **end for**
37: $\quad$ **for** $i \in P'(h)$ **do** $\triangleright$ Similar to line 18, we must perform these updates for all moving players
38: $\quad\quad$ **for** $I \in \vec{I}_i$ **do**
39: $\quad\quad\quad$ **for** $a \in A(I)$ **do**
40: $\quad\quad\quad\quad r_I[a] \leftarrow \max(r_I[a] + \vec{m}_i[a][I] - \vec{u}_i[I], 0)$
41: $\quad\quad\quad\quad s_I[a] \leftarrow s_I[a] + \vec{\pi}_i[I]\vec{\sigma}_i[I][a]w$
42: $\quad\quad\quad$ **end for**
43: $\quad\quad$ **end for**
44: $\quad$ **end for**
45: $\quad$ **return** $\vec{u}_{1...p}$
46: **end procedure**

Algorithm 2 Terminal value calculation

procedure TERMINALCFVS($h, \mathbf{b}, \vec{\pi}_{1...p}$)
    $\vec{v}_{1...p}[\cdot] \leftarrow 0$ ▷ Initialize factual values
    $\mathbf{b}_{\text{term}} \leftarrow$ CALCTERMINALBELIEF($h, \mathbf{b}, \vec{\pi}_{1...p}$)
    for $i = 1$ to p **do**
        for $\rho \in \mathbf{b}_{\text{term}}$ **do**
            $\vec{v}_i[I_i(h,\rho)] \leftarrow \vec{v}_i[I_i(h,\rho)] + \mathbf{b}_{\text{term}}[\rho]u_i(h,\rho)$
        end for
    end for
    return $\vec{v}_{1...p}/\vec{\pi}_{1...p}$ ▷ Convert factual to counterfactual
end procedure

procedure NEURALCFVS($h, \mathbf{b}, \vec{\pi}_{1...p}$)
    $\mathbf{b}_{\text{term}} \leftarrow$ CALCTERMINALBELIEF($h, \mathbf{b}, \vec{\pi}_{1...p}$)
    $w \leftarrow \sum_\rho \mathbf{b}_{\text{term}}[\rho]$
    $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_p \leftarrow \text{NN}[h](h, \mathbf{b}_{term}/w)$ ▷ Call NN with normalized belief
    return $w\vec{v}_{1...p}/\vec{\pi}_{1...p}$ ▷ Convert factual to counterfactual
end procedure

procedure CALCTERMINALBELIEF($h, \mathbf{b}, \vec{\pi}_{1...p}$)
    for $\rho \in \mathbf{b}$ **do**
        $\mathbf{b}_{\text{term}}[\rho] \leftarrow \mathbf{b}[\rho]\prod_i \vec{\pi}_i(I_i(h,\rho))$
        $\mathbf{b}_{\text{term}}[\rho] \leftarrow \mathbf{b}_{\text{term}}[\rho](1 - \mathbb{1}\{h \vdash \neg\rho\})$ ▷ Zero beliefs that are logically inconsistent
    end for
    return $\mathbf{b}_{\text{term}}$
end procedure

Algorithm 3 Backwards training

INPUT $P_{1...n}$: Dependency-ordered list of game parts.
INPUT $\Theta_{1...n}$: For each game part, a distribution over game situations.
INPUT d: The number of training datapoints generated per game partition.
OUTPUT $N_{1...n}$: n trained neural value networks, one for each game part.

procedure ENDTOENDTRAIN($P_{1...n}, \Theta_{1...n}, d$) ▷ Train a neural network for each game partition
    for $i = 1$ to n **do**
        $\mathbf{x}, \mathbf{y} \leftarrow$ GENERATEDATAPOINTS($P_i, \Theta_i, N_{1...i-1}$)
        $N_i \leftarrow$ TRAINNN($\mathbf{x}, \mathbf{y}$)
    end for
    return $N_{1...n}$
end procedure

procedure GENERATEDATAPOINTS($d, S, \Theta, N_{1...k}$) ▷ Given a game partition, it's distribution over game situations, and the NNs needed to limit solution depth, generate d datapoints.
    for $i = 1$ to d **do**
        $\theta_i \sim \Theta$ ▷ Sample a game situation from the distribution
        $\mathbf{v}_i \leftarrow$ SOLVESITUATION($S, \theta_i, N_{1...k}$) ▷ Solve that game situation for every player's values, using previously trained neural networks to solution depth.
    end for
    return $\theta_{1...d}, \mathbf{v}_{1...d}$ ▷ Return all training datapoints
end procedure

Algorithm 4 Game Situation Sampler

```
INPUT s: The number of succeeds.
INPUT f: The number of fails.
OUTPUT p, b: A random game situation from this game part, consisting of a proposer and a belief over the roles.

procedure SAMPLESITUATION(s, f)
    I ← SAMPLEFAILEDMISSIONS(s, f)        ▷ Uniformly sample f failed missions
    E ← EVILPLAYERS(I)                    ▷ Calculate evil teams consistent with the missions
    P(E) ~ Dir(1⃗_{|E|})                   ▷ Sample probability of each evil team
    P(M) ~ Dir(1⃗_n)                       ▷ Sample probability of being Merlin for all players
    b ← P(E) ⊗ P(M)                       ▷ Create a belief distribution using P(E) and P(M)
    p ~ unif{1, n}                        ▷ Sample a proposer uniformly over all the players
    return p, b
end procedure
```

4. 总结

创新亮点:

1. DeepRole算法中博弈树上的历史h不仅包括可见动作a，也可以包括观测结果o，以解决Avalon任务阶段队内投票各玩家只知计票结果，不知具体动作的问题。观测结果o到具体动作的映射，由逻辑推理模块完成。

2. DeepRole允许考虑同一个信息集上多个玩家需要决策的情况，以适应Avalon中的投票。

实验效果:

效果拔群。