论文阅读: NeruIPS 17's Best Paper

Safe and Nested Subgame Solving for Imperfect Information Games

金政羽

Clovers2333@gmail.com

浙江大学计算机科学与技术学院

2025 年 **5** 月 **4** 日

Contents

Head's up no-limit Texas hold'em

德州扑克 (Texas hold'em) 是一种流行的纸牌游戏，本文研究的是其中的单挑无限注形式 (Heads-up no-limit).

基本规则:

- 每位玩家获得两张底牌 (hole cards)，仅自己可见.
- 游戏分为四个下注阶段: 翻牌前 (preflop)、翻牌圈 (flop)、转牌圈 (turn) 和河牌圈 (river).
- 先由荷官 (dealer) 发牌，每轮按顺时针顺序行动.
- 下注方式包括: 盖牌 (fold)、跟注 (call) 或加注 (raise)、全下 (all-in).
- 无限注 (no-limit) 形式允许玩家在最低加注额基础上加注任意筹码.

游戏流程:

1. 发两张底牌给每位玩家，进行第一轮下注.
2. 翻牌: 发三张公共牌，进行第二轮下注.
3. 转牌: 发第四张公共牌，进行第三轮下注.
4. 河牌: 发第五张公共牌，进行最后一轮下注.
5. 若有多个玩家未盖牌，则亮牌比大小，按牌型分配彩金.

Imperfect-Information Games

- 不完美信息博弈 (Imperfect-information games) 模拟了具有隐藏信息的战略环境，它们有广泛的应用，包括谈判、拍卖、网络安全和物理安全.
- 在完美信息博弈中，确定一个决策点的最优策略只需要知道博弈树的当前节点以及该节点之后的剩余博弈树（即以此节点为根的子博弈）.
- 这一事实已被几乎所有用于完美信息博弈的人工智能所利用，包括击败顶级人类棋手的 AI（如国际象棋和围棋．在跳棋 (checkers) 中，甚至可以通过将博弈分解为独立的较小子博弈来完全解决整个博弈.

Imperfect-Information Games

- 然而，在不完美信息博弈中，仅使用该子博弈的知识是无法确定一个子博弈的最优策略的，因为博弈树的确切节点通常是未知的，最优策略可能取决于对手在某个未到达的子博弈中本可以获得的值.
- 尽管这看起来违反直觉，我们将在第 2 节中提供一个演示.
- 过去处理不完美信息博弈的方法通常不依赖子博弈分解，而是一开始就解决整个博弈.
- 例如，单挑限注德州扑克 (heads-up limit Texas hold'em)，一个相对简单的扑克形式，拥有 10^{13} 个决策点，基本上是在没有分解的情况下被解决的.
- 然而，这种方法无法扩展到更大的博弈，例如单挑无限注德州扑克 (heads-up no-limit Texas hold'em) ——它拥有 10^{161} 个决策点.

Imperfect-Information Games

在本节中，我们提供直觉说明为何在不完美信息博弈中子博弈求解不能孤立进行．我们将使用一个简单的 Coin Toss 游戏，如下图所示．

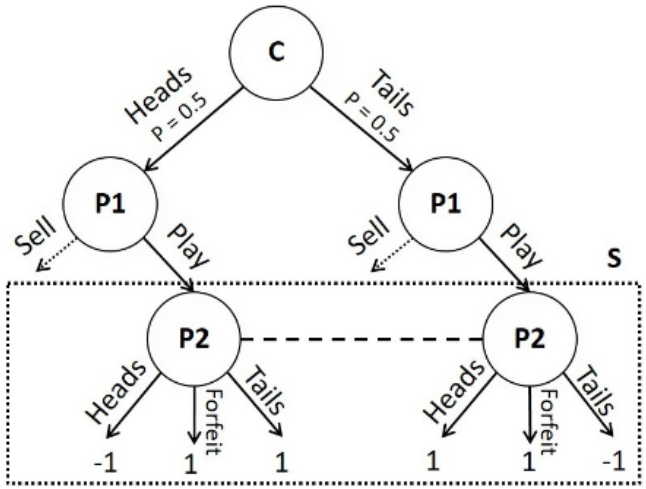


图: Coin Toss 游戏示意图

Coin Toss Example

游戏规则：

- 游戏由玩家 P1 和 P2 进行．P1 掷硬币，结果为 Heads 或 Tails，概率各为 0.5．
- P1 观察结果，但 P2 不观察．
- P1 然后选择一个动作：Play 或 Sell．
- 如果 P1 选择 Sell，游戏结束．如果硬币是 Heads，P1 赢得 $0.5；如果是 Tails，P1 损失 $0.5．
- 如果 P1 选择 Play，P2 必须猜测硬币的结果 (Heads 或 Tails)．
  - 如果 P2 猜对了，P1 支付 P2 $1．
  - 如果 P2 猜错了，P2 支付 P1 $1．
- Play 动作导致了一个 P2 进行猜测的子博弈．

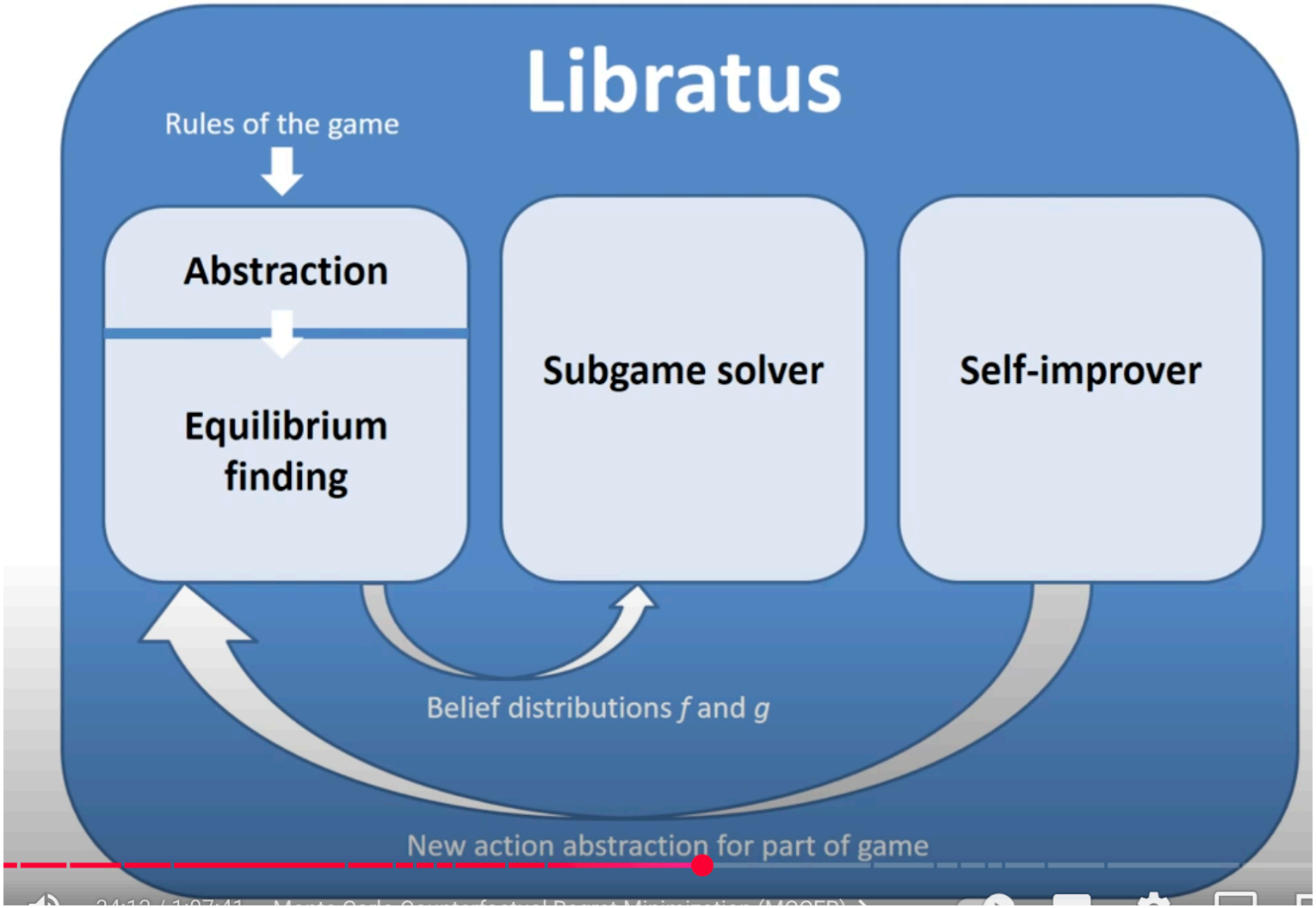
Coin Toss Example

回顾之前的完美信息博弈，在博弈树上，当我们到达了一个节点以后，我们根本无需关心这个节点之前发生的事情，只需要求解节点之后的博弈这个子问题即可，所以问题规模会不断缩小.

然而，在例如 Coin Toss 这样的非完美信息博弈中，我们在信息集中必须” 向上看”——利用后验概率判断对手的底牌，但是在传统的计算中父亲节点的决策显然是依赖于子节点均衡的收益的

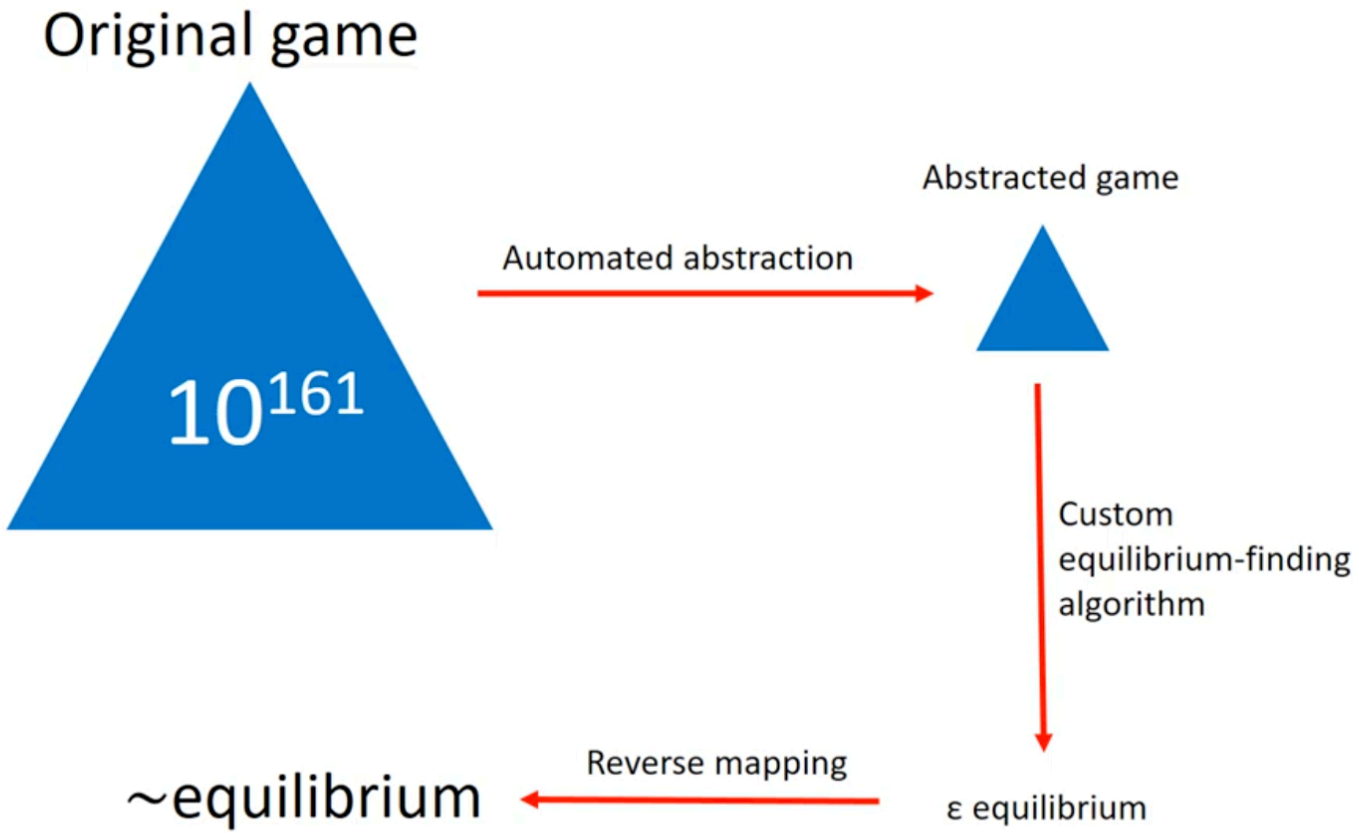
这样互相依赖的关系，会导致我们不能确定父子节点计算顺序——我们只能整棵树一起计算.

Outline of Libratus



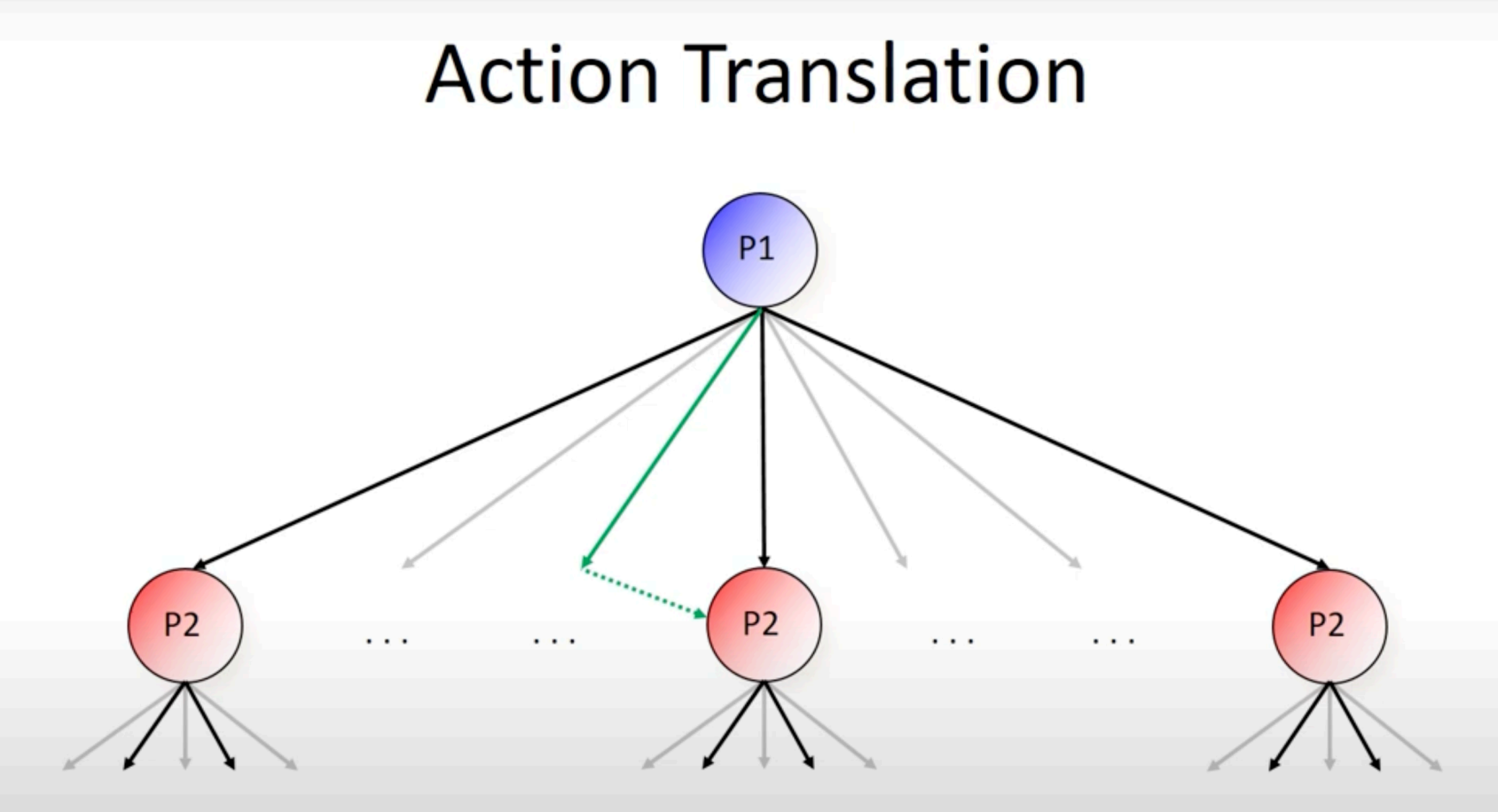
Abstraction

Abstraction [Gilpin & Sandholm EC-06, *J. of the ACM* 2007…]

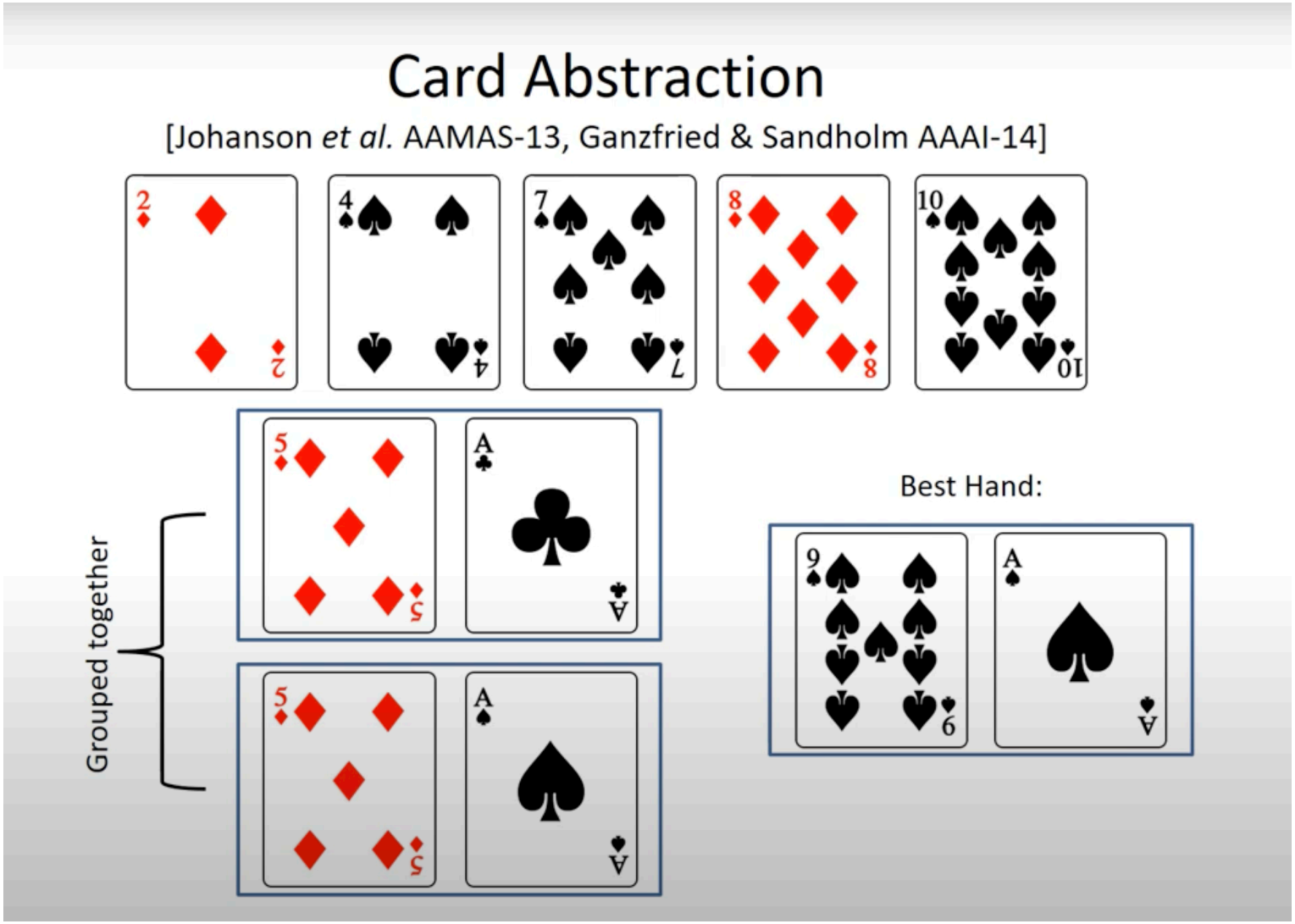


Foreshadowed by Shi & Littman 01, Billings *et al.* IJCAI-03

Abstraction



Abstraction



Extensive-Form Game

一个扩展式博弈定义为:

- 历史/节点 **(Histories/Nodes)** H**:** 有限节点集, 构成博弈树. 包含非终结节点和终结节点 $Z \subseteq H$. 根节点是空历史.
- 行动玩家 **(Player Function)** $P : H \setminus Z \to \{1, ..., n, c\}$**:** 指定在非终结节点 h 行动的玩家或机会玩家 c.
- 动作 **(Actions)** $A(h)$**:** 节点 h 可选的有限动作集合. $A(h) = \emptyset \iff h \in Z$.
- 后继节点 **(Successor Function):** 对 $h \in H \setminus Z, a \in A(h)$, $h \cdot a$ 是采取动作 a 后到达的节点.
- 机会概率 **(Chance Probabilities)** $p(h, a)$**:** 若 $P(h) = c$, 则 $p(h, \cdot)$ 是 $A(h)$ 上的概率分布, $\sum_{a \in A(h)} p(h, a) = 1$.
- 玩家收益 **(Player Payoffs)** $u_i : Z \to \mathbb{R}$**:** 玩家 i 在终结节点 z 的收益.

Imperfect Information and Strategy

Imperfect Information:

- 信息集 **(Infosets)** $\mathcal{I}_i$**:** 玩家 i 节点集 $H_i = \{h \in H | P(h) = i\}$ 的划分. 同一信息集 $I \in \mathcal{I}_i$ 中的节点 $h, h' \in I$ 对玩家 i 不可区分.
- 要求: $P(h) = P(h') = i$ 且 $A(h) = A(h')$. 记作 $A(I)$.
- 完美信息博弈: 每个信息集只含一个节点.

策略 **(Strategy):**

- (行为) 策略 σ_i: 映射, 为每个 $I \in \mathcal{I}_i$ 指定 $A(I)$ 上的概率分布 $\sigma_i(I)$.
- $\sigma_i(I, a)$: 在 I 选择动作 $a \in A(I)$ 的概率, $\sum_{a \in A(I)} \sigma_i(I, a) = 1$.
- 策略组合 $\sigma = (\sigma_1, ..., \sigma_n)$.
- σ_{-i}: 除玩家 i 外所有玩家的策略组合. $\sigma = (\sigma_i, \sigma_{-i})$.

Reach Probability and Expected Value

到达概率 **(Reach Probability):**

- $\pi^\sigma(h)$: 策略 σ 下到达节点 h 的总概率.
- $\pi_i^\sigma(h)$: 玩家 i 对到达 h 的贡献概率 (路径上玩家 i 动作概率乘积).
- $\pi_{-i}^\sigma(h)$: 除玩家 i 外所有玩家 (含机会) 对到达 h 的贡献概率.
- $\pi^\sigma(h) = \pi_i^\sigma(h)\pi_{-i}^\sigma(h)$.
- $\pi^\sigma(I) = \sum_{h\in I}\pi^\sigma(h)$: 到达信息集 $I \in \mathcal{I}_i$ 的概率.
- $\pi^\sigma(h|I) = \frac{\pi^\sigma(h)}{\pi^\sigma(I)}$: 到达信息集 I 时处于节点 h 的条件概率 (信念, 需 $\pi^\sigma(I) > 0$).

期望收益 **(Expected Value):**

- $v_i(h,\sigma)$: 从节点 h 开始, 按 σ 进行游戏, 玩家 i 的期望收益.
- $v_i(\sigma) = v_i(\text{root},\sigma) = \sum_{z\in Z}\pi^\sigma(z)u_i(z)$: 游戏总期望收益.
- $v_i(I,\sigma) = \sum_{h\in I}\pi^\sigma(h|I)v_i(h,\sigma) = \frac{\sum_{h\in I}\pi^\sigma(h)v_i(h,\sigma)}{\pi^\sigma(I)}$: 信息集 $I \in \mathcal{I}_i$ 上的期望收益 (需 $\pi^\sigma(I) > 0$).

Nash Equilibrium and Related Concepts

最佳响应 (Best Response - BR)

策略 σ_i^* 是对 σ_{-i}^* 的最佳响应, 如果:

$$\sigma_i^* \in BR(\sigma_{-i}^*) = \arg\max_{\sigma_i'} v_i(\sigma_i', \sigma_{-i}^*)$$

可利用性 (Exploitability)

衡量策略 σ 偏离 NE 的程度 (双人零和博弈为例):

$$\begin{aligned} expl(\sigma) &= \max_{\sigma_1'} v_1(\sigma_1', \sigma_2) + \max_{\sigma_2'} v_2(\sigma_1, \sigma_2') \\ &= \max_{\sigma_1'} v_1(\sigma_1', \sigma_2) - \min_{\sigma_2'} v_1(\sigma_1, \sigma_2') \\ &= (v_1(BR(\sigma_2), \sigma_2) - v_1(\sigma_1, \sigma_2)) \\ &\quad + (v_2(\sigma_1, BR(\sigma_1)) - v_2(\sigma_1, \sigma_2)) \end{aligned}$$

是每个玩家切换到 BR 的收益改进量之和. NE 的可利用性为 0.

Counterfactual Value

动作反事实值 (Counterfactual Value of an Action)

信息集 $I \in \mathcal{I}_i$ 中动作 $a \in A(I)$ 的反事实值:

$$v^{\sigma}(I,a) = \frac{\sum_{h \in I} \pi^{\sigma}_{-i}(h) v_i(h \cdot a, \sigma)}{\sum_{h \in I} \pi^{\sigma}_{-i}(h)}$$

采取动作 a 后，在后继节点 $h \cdot a$ 开始游戏的期望收益 (用 $\pi^{\sigma}_{-i}(h)$ 加权).

CFR Related Concepts and Subgames

反事实最佳响应 (Counterfactual Best Response - CBR)

信息集 $I \in \mathcal{I}_i$ 对 σ 的 CBR 动作是最大化 $v^\sigma(I,a)$ 的动作 $a^* \in A(I)$.

$$CBR^\sigma(I) = \arg\max_{a \in A(I)} v^\sigma(I,a)$$

反事实最佳值 (Counterfactual Best Value - CBV)

玩家 i 在信息集 $I \in \mathcal{I}_i$ 能达到的最大反事实值:

$$CBV^\sigma(I) = \max_{a \in A(I)} v^\sigma(I,a)$$

CFR Related Concepts and Subgames

不完美信息子博弈 (Imperfect-Information Subgame)

节点集合 $S \subseteq H$ 定义了一个子博弈, 如果满足:

1. 后继闭包 **(Successor Closure):** 若 $h \in S, h \notin Z$，则 $\forall a \in A(h), h \cdot a \in S$.
2. 信息集闭包 **(Infoset Closure):** 若 $h \in S$ 且 $h \in I \in \mathcal{I}_i$，则 $I \subseteq S$.

$S_{top} = \{h \in S \mid \text{父节点}(h) \notin S\}$ 是子博弈的入口节点.

Introduction to Subgame Solving

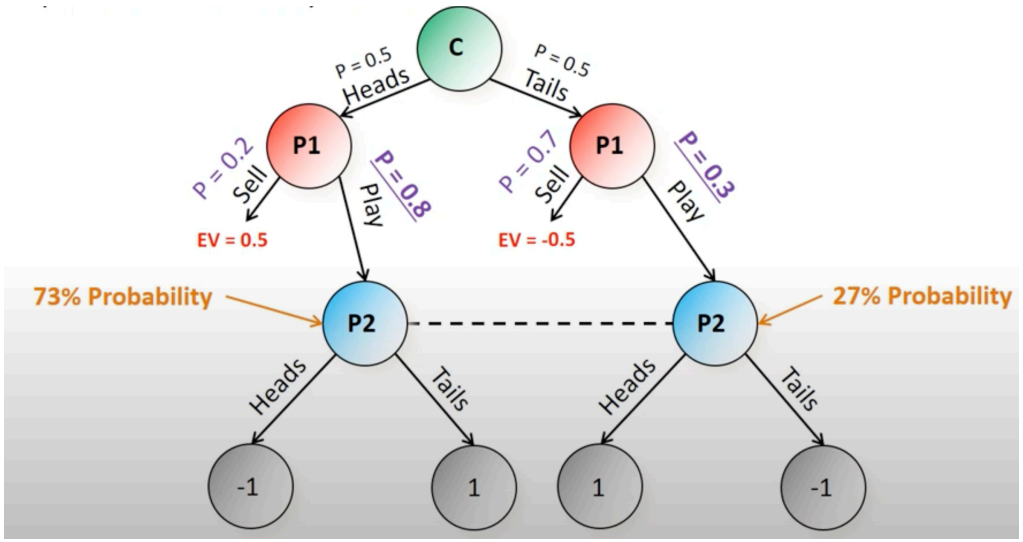
当游戏进行到一定阶段时，游戏的剩余规模已经支持我们进行在线计算，对于当前的博弈，重新解出一个更好的策略.

具体来说，假设之前的蓝图策略是 σ，我们希望在子博弈 S 中找到一个更好的策略 σ^S，使得 σ^S 在 S 之外大部分与 σ 相同，但在 S 内部使用计算出的 σ^S.

Unsafe Subgame Solving

一种最简单的想法是，我们直接非常暴力地假设对手在子博弈外采用的就是蓝图策略，然后我们在这个假设下求解子博弈.

很显然地，如果对手知道我们的蓝图策略，那么 Unsafe Subgame Solving 会变得很不成功：



Unsafe Subgame Solving

并且，Unsafe Subgame Solving 的进入特定节点的后验概率是基于之前蒙特卡洛算法得出的蓝图策略的结果，如果一上来，对手直接 all in 怎么办?

这种情况显然是非常少的，我们并没有在蒙特卡洛中对其进行详细的搜索，所以我们怎么知道后验概率呢？——完全随机？这显然是一个 bad idea.

Unsafe Subgame Solving

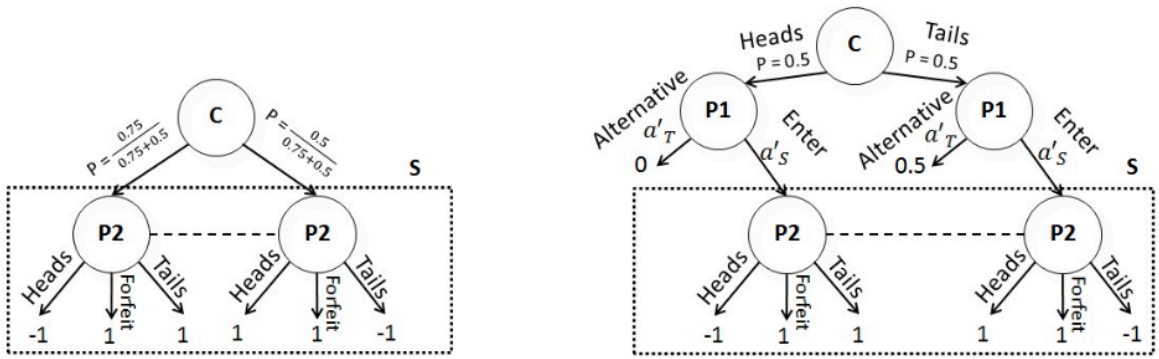
事实上，得益于简单的模型，并且对手并不知道蓝图策略，所以 Unsafe Subgame Solving 在实践中效果还不错.

但是归根到底，Subgame Solving 有几率增加对手的可利用性，使得我们的情况更劣. 从理论上来讲，我们希望找到一个更稳定的方法.

Safe Subgame Solving

针对以上两个问题，我们提出两种不同的方法.

如何确保求解新的子博弈时，对手对我们的策略的利用性不会增加? 我们可以在求解子博弈的时候，对于对手的每个排面进行分类，新添加节点，然后给新添加的节点多一个”退出机制”:



(a) Unsafe subgame solving (b) Resolve subgame solving

Figure 3: The augmented subgames solved to find a P_2 strategy in the Play subgame of Coin Toss.

Safe Subgame Resolving

对于我们当前决策的信息集 S_{top}，我们按照对手的排面进行分类，添加新点，根节点到新点到概率根据蓝图策略和题目背景计算，新点可以选择进入子博弈，也可以选择退出，直接获得当前信息集的 CBV.

然后对于这个新的博弈，我们重新跑纳什均衡求解，得到新的策略.

我们声称新的策略在原问题的任何一种对手牌面都不会劣于蓝图策略——如果有一个牌面劣于，对手可以在那个排面选择新的 best response，剩下的选择直接退出.

因此，在缩小问题规模且不会比原策略更差的前提下，我们可以更详细地跑一遍纳什均衡求解，得到新的策略.

Maxmargin Subgame Solving

另外一种方式，我们希望我们的新策略能在最坏情况下削减对手的收益.

Maxmargin 子博弈求解通过为每个入口信息集 $I_1 \in S_{top}$ 定义一个边界(margin)，以此来提高性能:

$$M^{\sigma^S}(I_1) = CBV^{\sigma_2}(I_1) - CBV^{\sigma_2^S}(I_1)$$

然后最大化最小边界值:

$$\min_{I_1 \in S_{top}} M^{\sigma^S}(I_1)$$

从定义就可以看到，这个方法一定是 safe 的.

Reach Subgame solving

在介绍新方法前，我们首先引入一个 gift 函数，用来描述对手在上一步所犯的错误的程度.

记 gift 函数 $g(I_1, a) = CBV(I_1) - CBV(I_1, a)$，表示对手在信息集 I_1 选择动作 a 而比它如果完美决策少赚的收益.

有了 gift 函数，我们可以修改之前的 Maxmargin 方法，定义 Reach margin：

$$M_r^{\sigma^S}(I_1) = M^{\sigma^S}(I_1) + \sum_{I_1', a' \in I_1 | P(I_1') = P_1} g(I_1', a')$$

我们可以将这个新函数运用到 Maxmargin 中，得到 Reach Maxmargin.

Reach Subgame solving

对于原先的 Maxmargin 方法，我们试图看出它的缺陷：

假设在当前的 Subgame，上一步对手做出了决策 a，如果对手手上拿着 $8,8$，那么他就犯错了，使得 CBV 下降了 \$100；但是如果他拿的是 $10,10$，那么他就没有犯错，CBV 并没有下降.

Maxmargin 的分析方法是，不管对手上一步做了什么，我们只关注现在的局面，要使得 CBV 尽可能稳定降低.

Reach Subgame solving

假设在蓝图策略中，上个局面下 8, 8 的 CBV 是 \$0，10, 10 的 CBV 是 \$200；那么在这个局面下 8, 8 的 CBV 是 \$-100，10, 10 的 CBV 是 \$200.

通过 Maxmargin 方法，我们使得这两种情况都 CBV 都稳定降低了 \$50.

但是，如果对手的策略是，他拿到 8, 8 时，根本不会进行行动 a，只有在拿到 10, 10 时才会进行行动 a，那么我们为了稳定 8, 8 而对 10, 10 决策作出的将就就显得有些浪费.

Reach Subgame solving 则通过引入 gift 函数，来避免这种浪费：通过将 gift 引入 Maxmargin 函数中，使得 8, 8 的 CBV 天生下降了 \$100，这样，我们或许可以作出使 10, 10 的 CBV 下降 \$75，而 8, 8 的 CBV 上升 \$25 的决策.

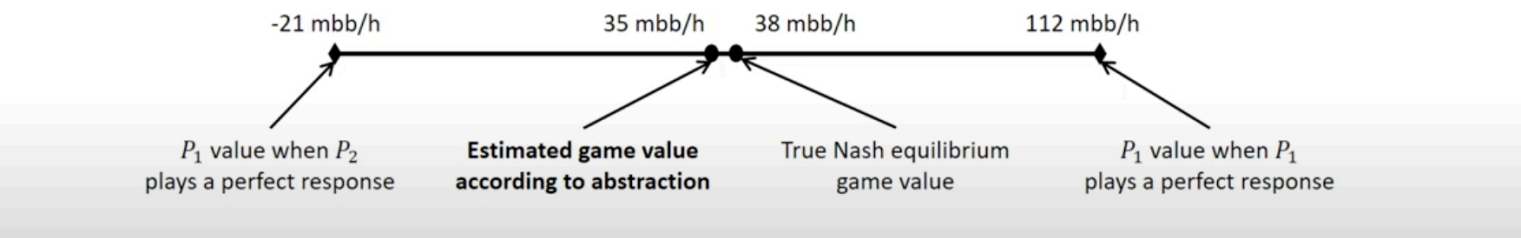
这样我们可以稳定赚 \$75，更加稳定.

Estimates for Alternative Payoffs

我们在 Reach Maxmargin 的方法中，我们使用对手针对蓝图策略 CBV 来估计对手之前的收益，这事实上是不准确的——我们的真正目的不是稳定地改善蓝图策略，而是希望稳定地减少对手的期望收益.

而在实验中，作者发现，如果双方都实用蓝图策略，那么最终的期望收益和双方都使用最优策略的期望收益是几乎相等的.

Test game of Flop Texas Hold'em using an abstraction that is 0.02% of the full game size:



Estimates for Alternative Payoffs

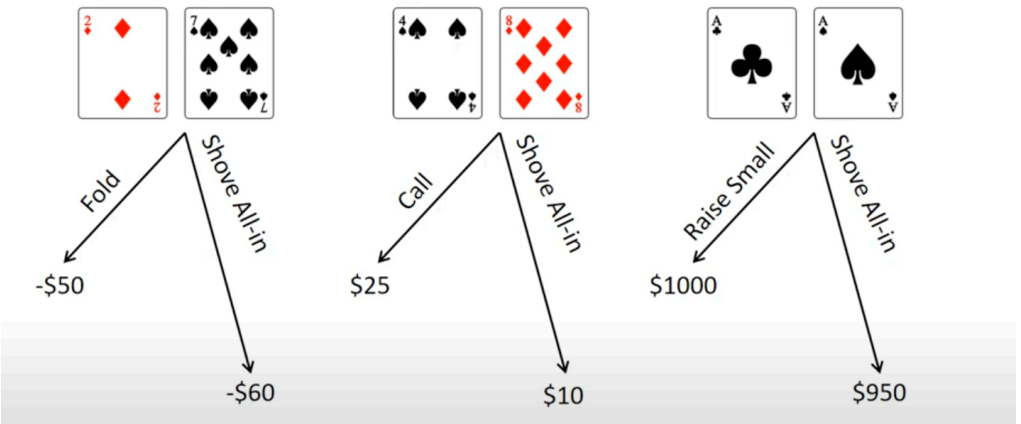
事实上，在有了上面的实验结果以后，我们可以证明我们用 Subgame Solving 得到的策略很接近纳什均衡：

定理

如果子博弈估计值接近真实值，那么子博弈求解得到的策略将接近纳什均衡策略.

Estimates for Alternative Payoffs

回过来看我们之前提出的对手突然 all in，但是我们样本不够的问题：



我们现在并不关心对手具体牌面的概率了，我们只需要知道，无论对手使用什么牌面，我们都可以尽可能减少他的期望收益.

Nested Subgame Solving

Nested Subgame Solving: 实时计算对手 off-tree 动作响应的技术，避免依赖预定义的转译.

- 核心思想: 当对手选择 off-tree 动作时，我们计算一个特定于该情况的响应，而不是使用一般的转译规则.
- 优势: 可以嵌套应用，对后续 off-tree 动作依次处理.
- 提出了两种主要方法：廉价方法 (inexpensive method) 和昂贵方法 (expensive method).

Nested Subgame Solving: Inexpensive Method

廉价方法 **(Inexpensive Method)**

1. 为该动作 a 后的子博弈 S 构建增强子博弈.
2. 对于子博弈中任何信息集 I_i，如果 P1 在信息集 I_i 中可能处于节点 h，则添加信息集 I_i 到子博弈.
3. 解决这个子博弈并得到新策略 σ^S.
4. 将 σ^S 与 σ 组合形成新蓝图 σ'，它在扩展的抽象中现在包含动作 a.

递归应用**:** 每当 P1 再次选择 off-tree 动作时，重复此过程.

廉价方法的局限**:** 无法与 Unsafe 子博弈求解结合，因为到达抽象外动作的概率未定义；也无法与 Reach Subgame solving 结合，因为礼物函数 $g(I_1, a)$ 未定义（蓝图策略未计算）.

Nested Subgame Solving: Expensive Method

昂贵方法 **(Expensive Method)**

- 基本思想: 如果我们希望结合 Unsafe 求解的优势，可以采用另一种方法.
- 实现方式: 在节点 h 而非 $h \cdot a$ 开始子博弈求解（其中 a 是 off-tree 动作）.
- 步骤:
  1. 在节点 h 构建包含此节点的最小子博弈.
  2. 将动作 a 添加到该抽象中.
  3. 对修改后的子博弈进行 Unsafe 求解.
- 缺点: 子博弈规模增大，因为必须为 $A(h)$ 中的每个动作（除了 a 外）重新计算策略.
- 极端情况: 如果对手在游戏开始时就选择了 off-tree 动作，则需要重新计算整个游戏策略.

Experiments

Small Flop Hold'em Flop Buckets:	**200**	**2,000**	**30,000**
Trunk Strategy	88.69	37.374	9.128
Unsafe	14.68	3.958	0.5514
Resolve	60.16	17.79	5.407
Maxmargin	30.05	13.99	4.343
Reach-Maxmargin	29.88	13.90	4.147
Reach-Maxmargin (not split)	24.87	9.807	2.588
Estimate	11.66	6.261	2.423
Estimate + Distributional	10.44	6.245	3.430
Reach-Estimate + Distributional	10.21	5.798	2.258
Reach-Estimate + Distributional (not split)	9.560	4.924	1.733

Table 1: Exploitability (evaluated in the game with no information abstraction) of subgame-solving in small flop Texas hold'em.

Experiments

Large Flop Hold'em Flop Buckets:	**200**	**2,000**	**30,000**
Trunk Strategy	283.7	165.2	41.41
Unsafe	65.59	38.22	396.8
Resolve	179.6	101.7	23.11
Maxmargin	134.7	77.89	19.50
Reach-Maxmargin	134.0	72.22	18.80
Reach-Maxmargin (not split)	130.3	66.79	16.41
Estimate	52.62	41.93	30.09
Estimate + Distributional	49.56	38.98	10.54
Reach-Estimate + Distributional	49.33	38.52	9.840
Reach-Estimate + Distributional (not split)	49.13	37.22	8.777

Table 2: Exploitability (evaluated in the game with no information abstraction) of subgame-solving in large flop Texas hold'em.

Experiments

Turn Hold'em Turn Buckets:	**200**	**2,000**	**20,000**
Trunk Strategy	684.6	465.1	345.5
Unsafe	130.4	85.95	79.34
Resolve	454.9	321.5	251.8
Maxmargin	427.6	299.6	234.4
Reach-Maxmargin	424.4	298.3	233.5
Reach-Maxmargin (not split)	333.4	229.4	175.5
Estimate	120.6	89.43	76.44
Estimate + Distributional	119.4	87.83	74.35
Reach-Estimate + Distributional	116.8	85.80	72.59
Reach-Estimate + Distributional (not split)	113.3	83.24	70.68

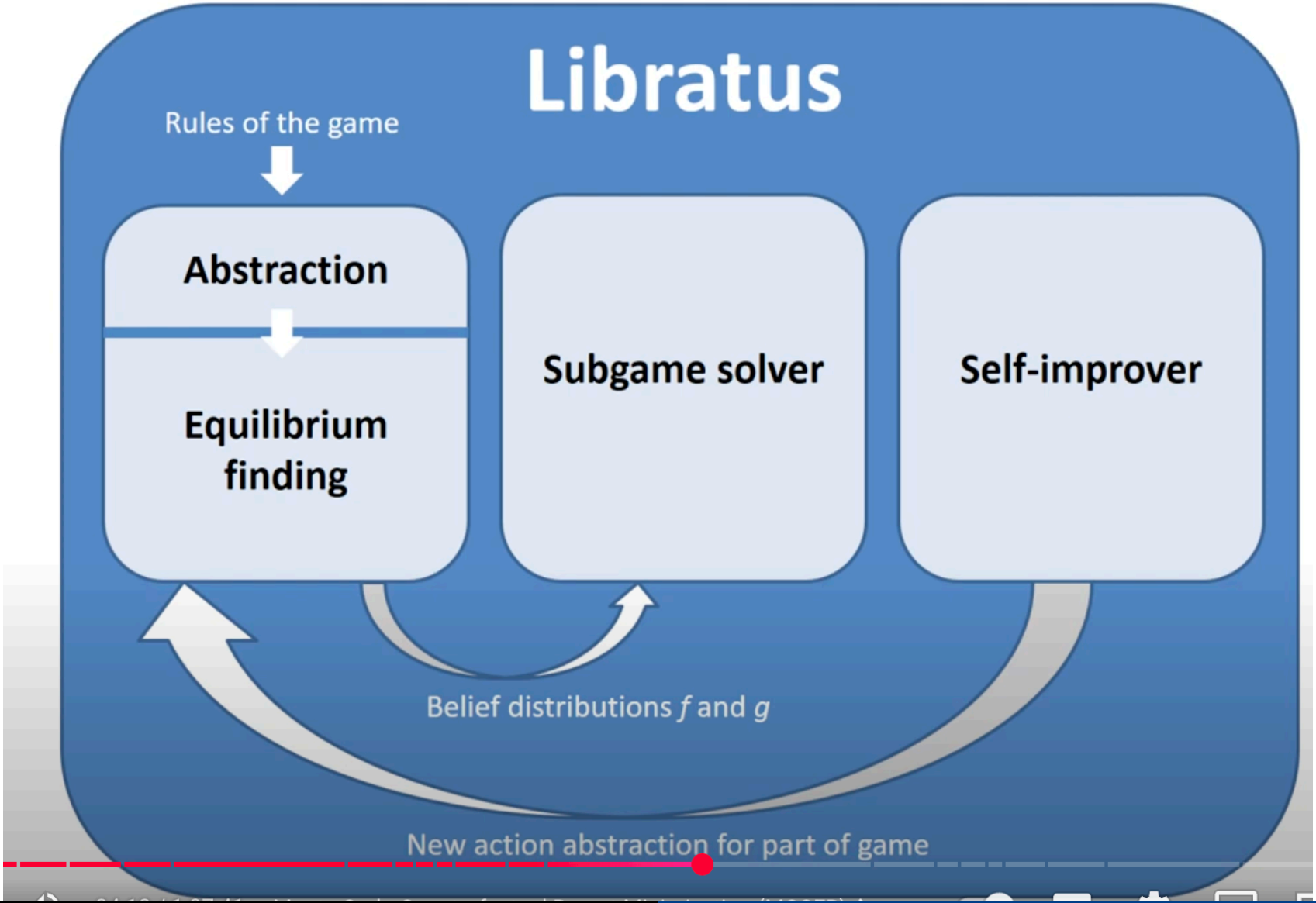
Table 3: Exploitability (evaluated in the game with no information abstraction) of subgame-solving in turn Texas hold'em.

Experiments

	mbb/h
Randomized Pseudo-Harmonic Mapping	1,465
Resolve	150.2
Reach-Maxmargin (Expensive)	149.2
Unsafe (Expensive)	148.3
Maxmargin	122.0
Reach-Maxmargin	119.1

Table 4: Exploitability of the various subgame-solving techniques in nested subgame solving. The performance of the pseudo-harmonic action translation is also shown.

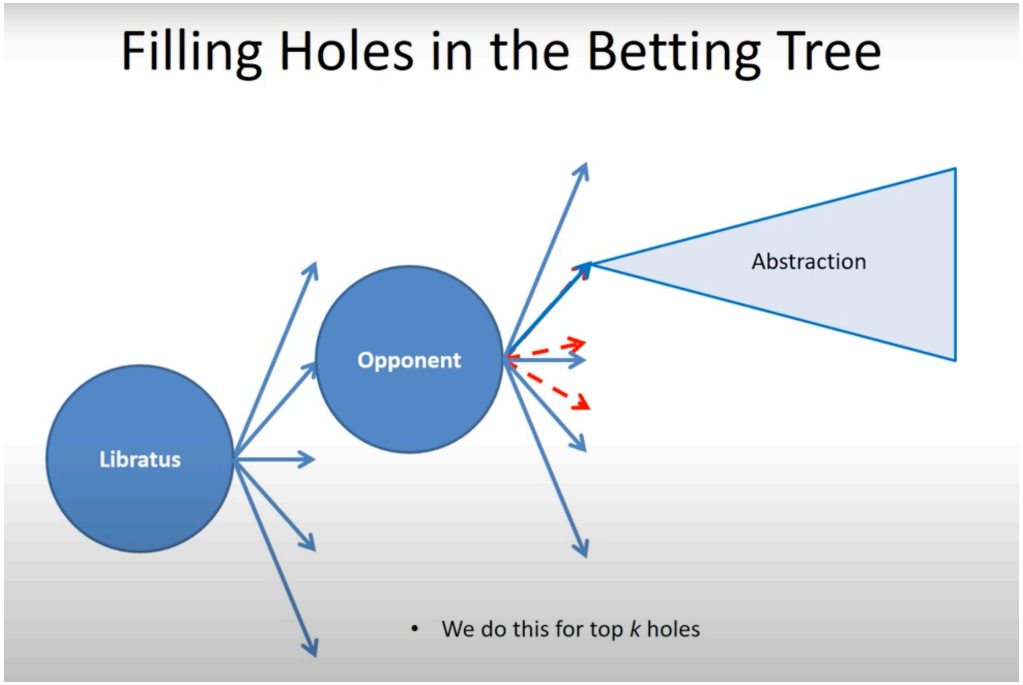
Self Improver



Self Improver

Self Improver 通过不断和自己对战、和玩家对战来修正蓝图策略.

简单来说，如果在对战当中有被对手利用率较高，且未在蓝图策略中被考虑的”Hole”，那么就会添加对于这个 Hole 的搜索.



Observation about Libratus's Play

Strengths:

- Many different bet sizes（更多下注组合）
- "Donk betting"（即使拿到差牌也可能会继续下注）
- Huge overbets （比人类更激进）
- Near-perfect balance（有时愿意舍弃沉没成本）

看上去第二个和第四个有点矛盾，但事实上都是在均衡下的选择．在某些情况下，均衡解可能就是通过一些不按寻常出牌的方式让对手拿到较为偏离的" 后验概率"，从而使得自己的获益最大．

Weaknesses:

- "No" opponent exploitation (几乎不利用对手弱点，仅仅遵循均衡策略)

What about 3+ players?

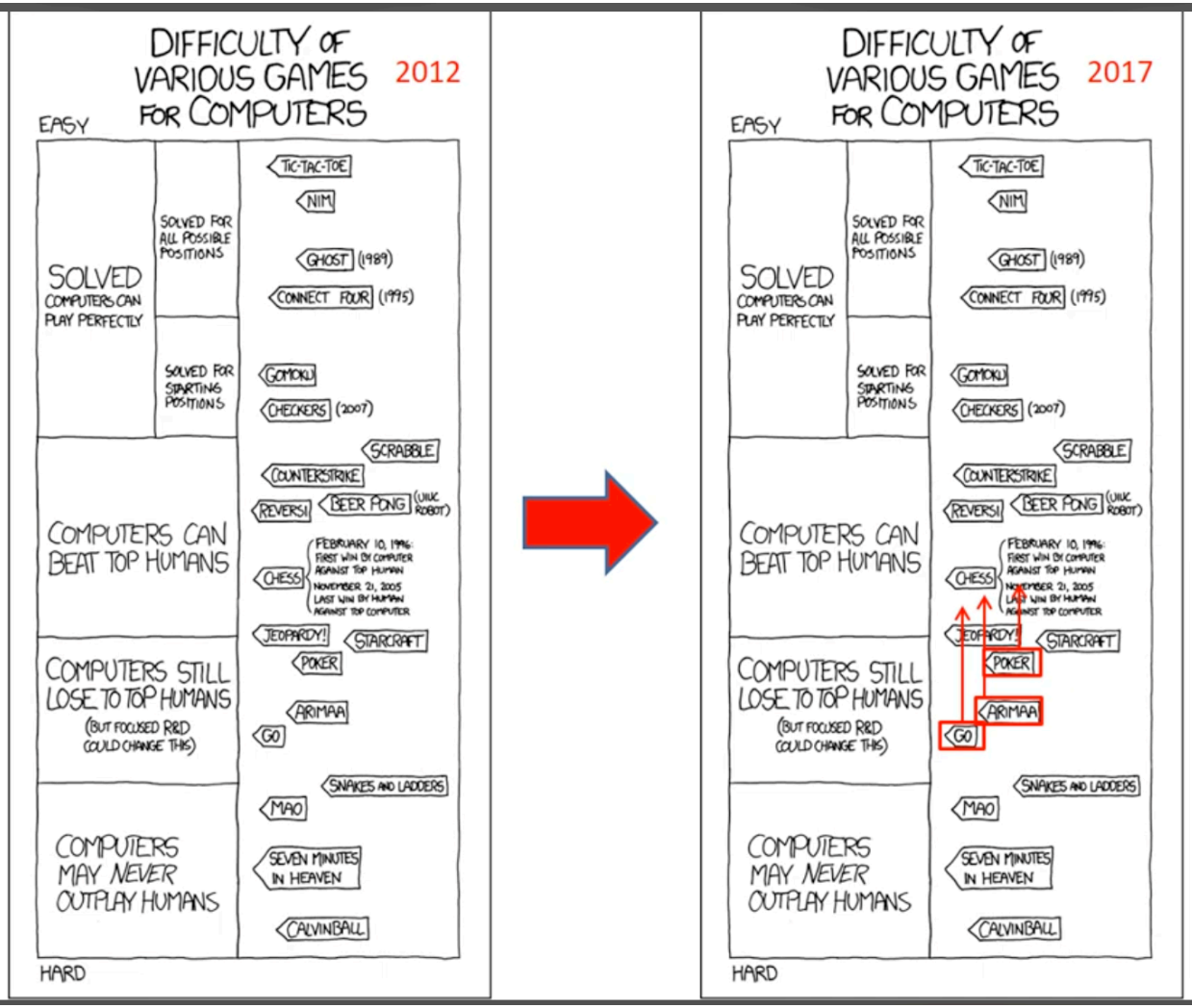
理论挑战：

- 计算纳什均衡成为 PPAD-complete 问题
- 3+ 玩家时，即使采用纳什均衡策略也可能在期望上亏损
- 表现依赖于对手之间的互动，难以建立客观的性能排名

实践情况：

- 相同技术在多人扑克中仍表现良好
- 大多数玩家会较早弃牌，使多人局迅速变为双人对抗
- 合作机会有限（且尝试合作违反规则）
- 目前几乎所有流行的扑克变种都有超越人类的 AI

Difficulties of various games for computer



Future Directions

博弈 **AI** 的未来研究方向:

- 博弈 **AI** 的” 大统一理论”
  - 开发能在超人类水平同时掌握围棋和扑克等不同类型博弈的单一算法
- 半合作博弈
  - 大富翁、风险 (Risk)、外交 (Diplomacy) 等多人战略游戏
- 现实世界应用
  - 金融市场
  - 谈判与协商
  - 安全领域
  - 拍卖机制

References

Safe and Nested Subgame Solving for Imperfect-Information Games, Noam Brown, Tuomas Sandholm, 2017.

Superhuman AI for heads-up no-limit poker: Libratus beats top professionals, Youtube.

Libratus: The Superhuman AI for No-Limit Poker, Noam Brown, Tuomas Sandholm, 2017.