Opponent-Limited Online Search for Imperfect Information Games

Weiming Liu [*1] **Haobo Fu** [*1] **Qiang Fu** [1] **Wei Yang** [1]

Abstract

In recent years, online search has been playing an increasingly important role in imperfect information games (IIGs). Previous online search is known as common-knowledge subgame solving, which has to consider all the states in a common-knowledge closure. This is only computationally tolerable for medium size games, such as poker. To handle larger games, order-1 Knowledge-Limited Subgame Solving (1-KLSS) only considers the states in a knowledge-limited closure, which results in a much smaller subgame. However, 1-KLSS is unsafe. In this paper, we first extend 1-KLSS to Safe-1-KLSS and prove its safeness. To make Safe-1-KLSS applicable to even larger games, we propose Opponent-Limited Subgame Solving (OLSS) to limit how the opponent reaches a subgame and how it acts in the subgame. Limiting the opponent's strategy dramatically reduces the subgame size and improves the efficiency of subgame solving while still preserving some safety in the limit. Experiments in medium size poker show that Safe-1-KLSS and OLSS are orders of magnitude faster than previous common-knowledge subgame solving. Also, OLSS significantly improves the online performance in a two-player Mahjong game, whose game size prohibits the use of previous common-knowledge subgame-solving methods.

1. Introduction

Artificial Intelligence (AI) has demonstrated superhuman performance in many large-scale Perfect Information Games (PIGs, e.g., Go and chess) and Imperfect Information Games (IIGs, e.g., Taxes hold'em) (Silver et al., 2016; 2017; Brown & Sandholm, 2018; Moravčík et al., 2017; Brown & Sandholm, 2019). In most AIs, online search has been critical in improving online performance. Take heads-up no-limit Texas hold'em as an example. A common approach is to first generate a *blueprint* strategy using Counterfactual Regret Minimization (CFR) algorithms (Zinkevich et al., 2007; Lanctot et al., 2009; Tammelin et al., 2015) and abstraction techniques (Gilpin & Sandholm, 2007; Gilpin et al., 2007). When playing online, subgame-solving techniques (Burch et al., 2014; Moravcik et al., 2016; Brown & Sandholm, 2017; Brown et al., 2018; Zhang & Sandholm, 2021) are used to refine the blueprint strategy, making it less exploitable and hopefully more profitable.

While it is straightforward to construct and solve a subgame for any state encountered online in PIGs, online search in IIGs is much more difficult. This is because one has to consider a subgame under *common knowledge* to keep the updated strategy *safe*, i.e., not more exploitable than the blueprint strategy. Prior approaches (Burch et al., 2014; Moravcik et al., 2016; Brown & Sandholm, 2017) usually enumerate the entire *common-knowledge closure*, which is the smallest set of states that the current state is in according to public information. Afterward, they construct and solve the *common-knowledge subgame*, which is a set of trees rooted at the states in the closure. For example, in two-player Texas hold'em, community cards and betting history are public information, and a common-knowledge closure contains around $\binom{52}{2} \times \binom{50}{2}$ states that enumerate every possible hole-card assignment of both players. These approaches work well in Texas hold'em but are computationally incapable of games with larger common-knowledge closure, e.g., dark chess.

The recently proposed Order-1 Knowledge-Limited Subgame Solving (1-KLSS) (Zhang & Sandholm, 2021) reduces the subgame size dramatically by only considering an *order-1 knowledge-limited subgame*, instead of the common-knowledge subgame. However, 1-KLSS is unsafe if applied independently to every state in the same common-knowledge closure (Zhang & Sandholm, 2021). Besides, the computational complexity of 1-KLSS is still unbearable in IIGs with large order-1 subgames, e.g., two-player Mahjong (Fu et al., 2022a) and Stratego (Perolat et al., 2022).

In this paper, we first revisit knowledge-limited subgame solving and propose Safe-1-KLSS. Specifically, we reformulate the common-knowledge subgame margin (Moravcik

[*]Equal contribution [1]Tencent AI Lab, Shenzhen, China. Correspondence to: Haobo Fu <haobofu@tencent.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

Algorithm	know.	safe	oppo.	subgame size	example
Unsafe	✗	✗	✓	$O(\|I^\infty\|)$	$O(N_h^2)$
Resolving	✗	✓	✗	$O(\|I^\infty\|)$	$O(N_h^2)$
Maxmargin	✗	✓	✗	$O(\|I^\infty\|)$	$O(N_h^2)$
Reach-Maxmargin	✗	✓	✗	$O(\|I^\infty\|)$	$O(N_h^2)$
1-KLSS	✓	✗	✗	$O(\|I^1\|)$	$O(N_h)$
Safe-1-KLSS (ours)	✓	✓	✗	$O(\|I^1\|)$	$O(N_h)$
OLSS-I (ours)	✓	✗	✓	$O(\|I^1\|)$	$O(N_h)$
OLSS-II (ours)	✓	✗	✓	$O(N)$	$O(N)$

Table 1: Overview of subgame solving. An algorithm is *knowledge-limited (know.)* if it performs in a knowledge-limited subgame. It is *safe* if it does not increase the exploitability. It is *opponent-limited (oppo.)* if it limits the opponent's strategy space. The "example" column gives the subgame size of a two-player Taxes hold'em with N_h hands. I^∞ is the common-knowledge closure and I^1 is the order-1 closure. N is the number of opponent strategies used in OLSS.

et al., 2016) and define a new term named *internal gift*. We show that a subgame solving is unsafe (safe) when the opponent's value is (not) allowed to increase more than the internal gift. Specially, 1-KLSS is unsafe as we show that 1-KLSS allows the opponent's value at every order-1 subgame to increase by the internal gift of the corresponding common-knowledge subgame. As a result, the total increase in the opponent's value can be much higher than the internal gift because a common-knowledge subgame corresponds to multiple order-1 subgames. In contrast, we propose Safe-1-KLSS, which distributes the internal gift to each order-1 subgame according to a reaching probability of the order-1 subgame. Moreover, we prove the safety of Safe-1-KLSS by showing that the total increase of the opponent's value is never allowed to exceed the internal gift.

Based on Safe-1-KLSS, we propose Opponent-Limited Subgame Solving (OLSS) to improve efficiency by limiting the strategy space of the opponent. We develop two types of OLSS. OLSS-I limits how the opponent reaches a subgame, i.e., the opponent can choose among some strategies for reaching the subgame. In this way, OLSS-I forces the player to focus more on the possible states that the opponent may reach, thus making the online search potentially more efficient. We prove that OLSS-I is safe when the opponent can choose any pure undominated strategy. Experiments in medium poker games show that Safe-1-KLSS and OLSS-I are orders of magnitude faster than previous common-knowledge subgame solving.

For even larger games that Safe-1-KLSS and OLSS-I are incapable of, we propose OLSS-II to further reduce the subgame size. OLSS-II only allows the opponent to choose among some strategies for the entire game. Thus, the agent is equivalently playing in a single-agent environment after the opponent has chosen its strategy. OLSS-II is unsafe when the number of opponent strategies is limited. Nevertheless, experiments in two-player Mahjong show that OLSS-II can significantly improve online performance by allowing the opponent to use only one or two strategies. An overview of previous and our subgame-solving algorithms is given in Table 1.

2. Notation

For a two-player zero-sum IIG, the set of players is denoted by $P = \{P_1, P_2\}$. Starting from the root $\emptyset$, the two players and chance (or nature) take actions in turn until a terminal node. The action history is denoted by h. The set of histories is denoted by H. For any non-terminal history $h \in H$, the set of legal actions at h is denoted by $A(h)$. The acting player at history h is denoted by $P(h)$, where $P(h) \in P \cup \{c\}$. c is chance, which chooses actions according to predefined distributions. After player $P(h)$ takes action $a \in A(h)$, the resulting history is denoted by ha. If a sequence of actions exists from h to h', then h' is a descendent of h, denoted by $h \sqsubset h'$. Let $h \sqsubseteq h'$ represent that $h \sqsubset h'$ or $h = h'$. For finite games we considered in this paper, the set of terminal histories is denoted by Z. For any $z \in Z$, the *payoff* for player p is $u_p(z) \in \mathbb{R}$. As the game is two-player and zero-sum, we have $u_1(z) = -u_2(z)$.

For each player, H can be divided into *information sets (infosets)* of the player. The set of infosets is denoted by $\mathcal{I}_p$. For any infoset $I \in \mathcal{I}_p$, any two histories $h, h' \in I$ are indistinguishable to p. Let $P(I) = P(h)$ and $A(I) = A(h)$ for any $h \in I$. Let $I_p(h)$ be the infoset of player p of history h. For two infosets $I, I' \in \mathcal{I}_p$, if there are $h \in I$ and $h' \in I'$ satisfy $h \sqsubseteq h'$, it is denoted by $I \sqsubseteq I'$.

A strategy of player $p \in P$ is denoted by $\sigma_p(I) \in \Delta(A(I))$ for $I \in \mathcal{I}_p, P(I) = p$, and $\sigma_p(I, a)$ is the probability of choosing action a. The set of strategies of player p is denoted by Σ_p. Since the histories in an infoset are indistinguishable, their strategies must be identical. So $\sigma_p(h, a) = \sigma_p(I, a)$ for any $h \in I$ when $P(I) = p$. The strategy of the other player and chance is denoted by σ_{-p}. A strategy profile $\sigma \in \Sigma$ is defined as $\sigma = \langle \sigma_p, \sigma_{-p} \rangle$. The chance player's strategy, which is fixed and known, is denoted by $\sigma_c(h, a)$.

$\pi^\sigma(h) = \prod_{h'a' \sqsubseteq h} \sigma_{P(h')}(h', a')$ is called the *reach* of h, which is the probability of reaching h when all the players act according to σ. $\pi_p^\sigma(h)$ is the contribution of p to this probability. $\pi_{-p}^\sigma(h)$ is the contribution of the opponent and chance. The probability of reaching h' from h is denoted

by $\pi^\sigma(h,h')$. Let $\pi^\sigma_p(I) = \pi^\sigma_p(h)$ for any $h \in I$, and define the reach of the opponent and chance as $\pi^\sigma_{-p}(I) = \sum_{h\in I}\pi^\sigma_{-p}(h)$. The *belief* $\beta^\sigma_{-p}(h|I) = \pi^\sigma_{-p}(h)/\pi^\sigma_{-p}(I)$ is the distribution of $h \in I$ given that all the players follow σ and I is reached.

The *expected payoff* of the game for player p is denoted by $u_p(\sigma_p,\sigma_{-p})$. Formally, $u_p(\sigma_p,\sigma_{-p}) = \sum_{z\in Z}\pi^\sigma(z)u_p(z)$. A *best response* $BR(\sigma_{-p})$ is a strategy of player p such that $u_p(BR(\sigma_{-p}),\sigma_{-p}) = \max_{\sigma'_p\in\Sigma_p} u_p(\sigma'_p,\sigma_{-p})$. A *Nash equilibrium* $\sigma^* = \langle\sigma^*_p,\sigma^*_{-p}\rangle$ is a strategy profile where every player plays a best response. The *exploitability* of a strategy σ_p is the distance to a Nash equilibrium, defined as $e(\sigma_p) = u_p(\sigma^*_p, BR(\sigma^*_p)) - u_p(\sigma_p, BR(\sigma_p))$. Define the total exploitability as $e(\sigma) = \sum_{p\in P} e(\sigma_p)$. When $e(\sigma) = \epsilon$, σ is called an ϵ-equilibrium.

Given a strategy σ, the *counterfactual value* at infoset I of player p is $v^\sigma_p(I) = \sum_{h\in I}\beta^\sigma_{-p}(h|I)u_p(h)$, where $u_p(h) = \sum_{z\in Z}\pi^\sigma(h,z)u_p(z)$ is the *expected value*. Let $v^\sigma_p(I,a) = \sum_{h\in I}\beta^\sigma_{-p}(h|I)u_p(ha)$. A *counterfactual best response* $CBR(\sigma_p)$ is a best response that maximizes the counterfactual value at every infoset. Define the *counterfactual best response value* as $CBV^{\sigma_{-p}}(I) = v_p^{\langle CBR(\sigma_{-p}),\sigma_{-p}\rangle}(I)$ and $CBV^{\sigma_{-p}}(I,a) = v_p^{\langle CBR(\sigma_{-p}),\sigma_{-p}\rangle}(I,a)$.

For a set of nodes denoted by S, let $\sigma^S \in \Sigma^S$ ($\sigma^S_p \in \Sigma^S_p$ for player p) be the strategy in S. Let $\sigma_{[S\leftarrow\sigma^S]}$ be the strategy that uses σ^S in S and σ elsewhere. Let S_{top} be the set of root nodes of S, i.e., the set of nodes in S whose parent nodes are not in. Define the *belief*, i.e., the distribution of $h \in S_{top}$ as $\beta^\sigma(h|S_{top}) = \pi^\sigma(h)/\sum_{h'\in S_{top}}\pi^\sigma(h')$ when all the players follow σ. Let $\mathcal{I}_p^{S_{top}} = \{I_p(h)|h \in S_{top}\}$ be the set of root infosets of player p. Define the *counterfactual value* for player p at S_{top} as $v^\sigma_p(S_{top}) = \sum_{h\in S_{top}}\beta^\sigma(h|S_{top})u^\sigma_p(h)$ when σ is used, and similarly define the *counterfactual best response value* as $CBV_p^{\langle\sigma_p,\sigma_{-p}\rangle}(S_{top}) = \sum_{h\in S_{top}}\beta^\sigma(h|S_{top})CBV^{\sigma_{-p}}(I_p(h))$.

3. Previous Subgame Solving

Subgame solving is a technique for refining strategies online for IIGs. In an IIG, instead of only considering the current infoset, it needs to analyze the *common-knowledge closure*, which is the smallest set of infosets that the current infoset is in according to public information. Take the game shown in Figure 1 as an example, I_a and I_b are in the same common-knowledge closure, and their strategy should be considered simultaneously. This is because the strategy at I_a affects the strategy at I_c and then the strategy at I_b, and vice versa. We define infoset hypergraph and common-knowledge closure (Zhang & Sandholm, 2021) formally as follows:

Definition 3.1. The infoset hypergraph G of a game is a hypergraph whose vertices are histories in H, and whose hyperedges are information sets.

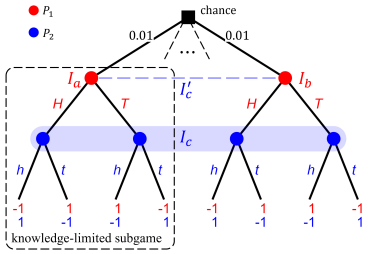


Figure 1: The 100-matching pennies. At the root, chance chooses an integer $n \in \{1,\dots,100\}$ randomly and uniformly, which is only visible to P_1. Then, the players play a game of matching pennies. The colored nodes indicate the players to move. I_a and I_b are P_1's infosets (the other 98 infosets are not shown). I_c and I'_c (the set of the nodes linked by the blue dotted line) are P_2's infosets. Inside the dotted box is an order-1 subgame.

Definition 3.2. For any $S \subseteq H$, the *order-k knowledge set* S^k is the set of histories that are at most distance $k-1$ away from S in G. The *common-knowledge closure* S^∞ is the connected component of G containing S.

Take I_a in Figure 1 as an example, we have $I_a^1 = I_a$ and $I_a^2 = I_a^\infty = I_a \cup \dots \cup I_b$. Given an infoset I, the *order-k subgame* $\overline{I^k}$ is a set of trees rooted at the histories in I^k. Specially, $\overline{I^\infty}$ is named the *common-knowledge subgame*. In this paper, we use S to denote the common-knowledge subgame $\overline{I^\infty}$ and use S_I to denote the order-1 subgame $\overline{I^1}$ for simplicity. Without loss of generality, we assume $P(h) = P_1$ for $h \in S_{top}$, and we refine the strategy in S or S_I for P_1. A common-knowledge subgame is a well-defined IIG given a distribution of $h \in S_{top}$.

3.1. Common-Knowledge Subgame Solving

For a subgame S, common-knowledge subgame solving updates the entire strategy of P_1 in S. The most intuitive one, known as *Unsafe solving* (Gilpin & Sandholm, 2006), assumes that all players follow the blueprint to reach the subgame. In other words, it assumes the distribution of $h \in S_{top}$ is exactly $\beta^\sigma(h|S_{top})$. Unsafe solving has been found to do extremely well in some cases (Brown & Sandholm, 2017; 2018; 2019; Moravčík et al., 2017). However, Unsafe solving can increase exploitability sometimes.

Resolving (Burch et al., 2014), *Maxmargin solving* (Moravcik et al., 2016), and *Reach-Maxmargin solving* (Brown & Sandholm, 2017) are safe subgame solving algorithms in the sense that they guarantee the *common-knowledge subgame margin* non-negative at every P_2 root infoset $I_2 \in \mathcal{I}_2^{S_{top}}$. This margin is defined as

$$M^\sigma_{CK}(I_2) = CBV^{\sigma_1}(I_2) - CBV^{\sigma_{1[S\leftarrow\sigma^S_1]}}(I_2). \quad (1)$$

It measures how much the opponent CBV decreases: if $M^{\sigma^S}_{CK}(I_2) \geq 0$ for each $I_2 \in \mathcal{I}_2^{S_{top}}$, the exploitability is guaranteed to not increase by the recursive nature of counterfactual values. Specifically, Resolving is an algorithm that tries to recover a strategy that satisfies $M^{\sigma^S}_{CK}(I_2) \geq 0, \forall I_2 \in \mathcal{I}_2^{S_{top}}$, while Maxmargin solving tries to maximize the minimum margin: $\max_{\sigma_1^S \in \Sigma_1^S} \min_{I_2 \in \mathcal{I}_2^{S_{top}}} M^{\sigma^S}_{CK}(I_2)$.

In practice, it may be too conservative to guarantee that $M^{\sigma^S}_{CK}(I_2) \geq 0$ at every $I_2 \in \mathcal{I}_2^{S_{top}}$. The reason is the values at some P_2's infosets may be so poor that the infosets should not be reached. In other words, if P_2 has reached such an infoset, it must have taken mistake actions in history. Reach-Maxmargin solving catches this intuition and uses a value g, named the *gift*, to estimate how much P_2 may have wasted to reach an infoset. The gift is then added to the margin, which results in the *reach-margin*: $M^{\sigma^S}_{R}(I_2) = M^{\sigma^S}_{CK}(I_2) + g(I_2)$. This margin is guaranteed non-negative in Reach-Maxmargin solving. In order to guarantee that the exploitability does not increase, the non-negative $g(I_2)$ should be carefully estimated. However, this gift is difficult to estimate in large IIGs.

3.2. Knowledge-Limited Subgame Solving

Common-knowledge subgame solving requires solving a common-knowledge subgame, which could be too large to solve online. k-KLSS (mostly $k = 1$) (Zhang & Sandholm, 2021) relieves this problem by fixing the strategy outside of the k-order subgame. So, the nodes outside $\overline{I^{K+1}}$ can be pruned as both players' strategies will not change there. Besides, the nodes in $\overline{I^{K+1}} \setminus \overline{I^k}$ can also be discarded since their values can be precomputed according to P_2's strategy and the blueprint. As a result, k-KLSS only needs to solve an order-k subgame $\overline{I^k}$, and any common-knowledge subgame-solving algorithm can be used to solve $\overline{I^k}$. In this paper, we assume the default solver is Maxmargin solving.

However, 1-KLSS is unsafe when applied to every infoset reached during play (Zhang & Sandholm, 2021). Take the game shown in Figure 1 as an example, and assume the blueprint strategy of P_1 is to choose H and T with probabilities 0.55 and 0.45. Applying 1-KLSS to I_a changes the strategy to choose T with probability 1, which reduces the exploitability. However, if it is applied to *every* infoset, P_1 will choose T everywhere, and the exploitability will increase from 0.1 to 1. The problem arises because the fixed strategy assumption is untrue in this case. The authors (Zhang & Sandholm, 2021) have proposed methods to obtain the safety guarantee by updating the blueprint or allocating deviations from the blueprint. However, these methods require much memory for recording additional information, which may be impractical in large IIGs. In the next section, we provide more analysis of knowledge-limited subgame solving and propose Safe-1-KLSS.

4. Safe Knowledge-Limited Subgame Solving

In this section, we first revisit the common-knowledge subgame margin and show why 1-KLSS is unsafe. Then, we propose Safe-1-KLSS, which decomposes a common-knowledge subgame margin into multiple safe-1-KLSS margins and guarantees safety by maximizing the safe-1-KLSS margins separately.

4.1. Revisiting Subgame Margin

As defined before, a *common-knowledge subgame margin* is $M^{\sigma^S}_{CK}(I_2) = CBV^{\sigma_1}(I_2) - CBV^{\sigma_1[S\leftarrow\sigma_1^S]}(I_2)$, which can be reformulated as

$$M^{\sigma^S}_{CK}(I_2) = \min_{\sigma_2^S \in \Sigma_2^S} \left\{ d^{\sigma^S}(I_2) + R^{\sigma_2^S}(I_2) \right\}, \text{where} \quad (2)$$

$$d^{\sigma^S}(I_2) := v_2^{\langle \sigma_1, \sigma_2^S \rangle}(I_2) - v_2^{\langle \sigma_1[S\leftarrow\sigma_1^S], \sigma_2^S \rangle}(I_2),$$

$$R^{\sigma_2^S}(I_2) := CBV^{\sigma_1}(I_2) - v_2^{\langle \sigma_1, \sigma_2^S \rangle}(I_2) \geq 0.$$

Note that the $v_2^{\langle \sigma_1, \sigma_2^S \rangle}(I_2)$ in $d^{\sigma^S}(I_2)$ and $R^{\sigma_2^S}(I_2)$ can be cancelled out and $\min_{\sigma_2^S \in \Sigma_2^S} -v_2^{\langle \sigma_1[S\leftarrow\sigma_1^S], \sigma_2^S \rangle}(I_2)$ is equivalent to $-CBV^{\sigma_1[S\leftarrow\sigma_1^S]}(I_2)$. So the $M^{\sigma^S}_{CK}(I_2)$ in Eq. (2) is exactly the common-knowledge subgame margin. Intuitively, we can consider $d^{\sigma^S}(I_2)$, which equals zero when $\sigma_1^S = \sigma_1$ in S, as the new objective that P_1 wants to maximize; and $R^{\sigma_2^S}(I_2)$, which is non-negative, as the *internal gift* to P_1. Similar to the gift in Reach-Maxmargin solving (Brown & Sandholm, 2017), internal gift allows $d^{\sigma^S}(I_2) \leq 0$ while it is still guaranteed that $M^{\sigma^S}_{CK}(I_2) \geq 0$.

The formulation allows us to further decompose the new objective into the belief-weighted sum of the sub-objective, denoted by $d^{\sigma^{S_I}}(I_2)$, in each order-1 subgame $I \in \mathcal{I}_1^{S_{top}}$:

$$d^{\sigma^S}(I_2) = \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta^{\sigma}_{-2}(I|I_2) d^{\sigma^{S_I}}(I_2), \text{where} \quad (3)$$

$$d^{\sigma^{S_I}}(I_2) := \sum_{h \in I \cap I_2} \beta^{\sigma}_{-2}(h|I \cap I_2) \left(u_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(h) - u_2^{\sigma^{S_I}}(h) \right).$$

The derivation is given in Appendix A. In the equation, $\beta^{\sigma}_{-2}(I|I_2) = \sum_{h \in I \cap I_2} \beta^{\sigma}_{-2}(h|I_2)$ is the probability of reaching I when I_2 is reached; and $\beta^{\sigma}_{-2}(h|I \cap I_2) = \beta^{\sigma}_{-2}(h|I_2) / \sum_{h \in I \cap I_2} \beta^{\sigma}_{-2}(h|I_2)$ is the probability to reach h when I and I_2 are reached. Note that both $\beta^{\sigma}_{-2}(I|I_2)$ and $\beta^{\sigma}_{-2}(h|I \cap I_2)$ are non-negative, and $\sum_{I \in \mathcal{I}_1^{S_{top}}} \beta^{\sigma}_{-2}(I|I_2) = 1$, $\sum_{h \in I \cap I_2} \beta^{\sigma}_{-2}(h|I \cap I_2) = 1$. So the margin $M^{\sigma^S}_{CK}(I_2)$ can be rewritten into

$$\min_{\sigma_2^S \in \Sigma_2^S} \left\{ \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta^{\sigma}_{-2}(I|I_2) d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^S}(I_2) \right\}. \quad (4)$$

The equation shows that the *belief-weighted sum* of $d^{\sigma^{S_I}}(I_2)$ should be greater than $-R^{\sigma_2^S}(I_2)$, in order to guarantee $M_{CK}^{\sigma^S}(I_2) \geq 0$. Recall that $R^{\sigma_2^S}(I_2) \geq 0$.

4.2. Revisiting 1-KLSS: Why is it Unsafe?

Now we formally define the optimization target of 1-KLSS and demonstrate why it is unsafe. Given a subgame S and an order-1 subgame S_I, 1-KLSS assumes that $\sigma_1^S(h) = \sigma_1(h)$ for $h \in S \setminus S_I$. So, $d^{\sigma^{S_{I'}}}(I_2) = 0$ for $I' \neq I$, and Eq. (4) can be rewritten into the *1-KLSS margin*:

$$M_{KL}^{\sigma^S}(I_2) = \min_{\sigma_2^S \in \Sigma_2^S} \left\{ \beta_{-2}^{\sigma}(I|I_2) d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^S}(I_2) \right\}. \tag{5}$$

The target of 1-KLSS is then to maximize the minimum 1-KLSS margin: $\max_{\sigma_1^{S_I} \in \Sigma_1^{S_I}} \min_{I_2 \in \mathcal{I}_2^{S_{top}}} M_{KL}^{\sigma^S}(I_2)$. 1-KLSS is safe if it is only applied to S_I and leaves the strategy in $S_{I'}, I' \neq I$ unchanged. However, if 1-KLSS is applied to *every* order-1 subgame in S independently, the sum of $\beta_{-2}^{\sigma}(I|I_2) d^{\sigma^{S_I}}(I_2)$ in Eq. (4) is likely to be less than $-R^{\sigma_2^S}(I_2)$. This is because Eq. (5) allows every $\beta_{-2}^{\sigma}(I|I_2) d^{\sigma^{S_I}}(I_2)$ to be as little as $-R^{\sigma_2^S}(I_2)$ while maximizing $M_{KL}^{\sigma^S}(I_2)$. Therefore, 1-KLSS can not guarantee that $M_{CK}^{\sigma^S}(I_2) \geq 0$ by maximizing $M_{KL}^{\sigma^S}(I_2)$ at every infoset $I \in \mathcal{I}_1^{S_{top}}$, and it is unsafe.

4.3. Safe-1-KLSS: Decomposing the Margin

Eq. (4) shows that the belief-weighted sum of $d^{\sigma^{S_I}}(I_2)$ should be greater than $-R^{\sigma_2^S}(I_2)$, which inspires us to distribute the gift $R^{\sigma_2^S}(I_2)$ to each order-1 subgame and decompose $M_{CK}^{\sigma^S}$, just like the decomposition of $d^{\sigma^S}(I_2)$ in Eq. (3). Specifically, according to Eq. (4) and Jensen's inequality, we prove that

$$M_{CK}^{\sigma^S}(I_2) \geq \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) M_{SKL}^{\sigma^{S_I}}(I_2), \text{where} \tag{6}$$

$$M_{SKL}^{\sigma^{S_I}}(I_2) := \min_{\sigma_2^{S_I} \in \Sigma_2^{S_I}} \left\{ d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^{S_I}}(I_2) \right\}. \tag{7}$$

The proof is given in Appendix A. In Eq. (7), $R^{\sigma_2^{S_I}}(I_2) = CBV^{\sigma_1}(I_2) - v_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(I_2)$ is the gift, and $M_{SKL}^{\sigma^{S_I}}(I_2)$ is named the *Safe-1-KLSS margin*. As we can see, $M_{CK}^{\sigma^S}$ is lower bounded by the belief-weighted sum of the Safe-1-KLSS margins. This motivates us to propose **Safe-1-KLSS**, which maximizes $M_{CK}^{\sigma^S}(I_2)$ by independently maximizing the minimum Safe-1-KLSS margin in each infoset $I \in \mathcal{I}_1^{S_{top}}$: $\max_{\sigma_1^{S_I} \in \Sigma_1^{S_I}} \min_{I_2 \in \mathcal{I}_2^{S_{top}}} M_{SKL}^{\sigma^{S_I}}(I_2)$.

Notice that $M_{SKL}^{\sigma^{S_I}}(I_2) = 0$ if $\sigma_1^{S_I} = \sigma_1$ in S_I, it is guaranteed that $M_{SKL}^{\sigma^{S_I}}(I_2) \geq 0$ after applying Safe-1-KLSS to subgame S_I. As a result, it is guaranteed that $M_{CK}^{\sigma^S}(I_2) \geq 0$ and Safe-1-KLSS is safe:

Theorem 4.1. *Given a strategy σ_1, a common-knowledge subgame S, a set of subgames $\mathbb{S} = \{S_I | I \in \mathcal{I}_1^{S_{top}}\}$, and a strategy $\sigma_1^{S_I}$ for each subgame S_I produced by applying Safe-1-KLSS, let σ_1' be the strategy that plays according to $\sigma_1^{S_I}$ for each subgame S_I and σ_1 elsewhere. If $\pi_1^{BR(\sigma_1')}(I_2) > 0$ for some $I_2 \in \mathcal{I}_2^{S_{top}}$, then $e(\sigma_1') \leq e(\sigma_1) - \sum_{h \in I_2} \pi_{-2}^{\sigma_1}(h) \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) M_{SKL}^{\sigma^{S_I}}(I_2)$.*

The proof is given in Appendix A. To solve an order-1 subgame using Safe-1-KLSS, we can construct a gadget game (Burch et al., 2014; Moravcik et al., 2016) and use CFR to compute the strategy. More details are provided in Section 6. Safe-1-KLSS can be performed at every infoset reached during play in a nested fashion. However, instead of constructing a new order-1 subgame every time a new infoset is reached, a new subgame should be extracted from the last gadget game in Safe-1-KLSS. This is because an order-1 subgame is incomplete in the sense of subgame solving (but a gadget game is well-defined).

While both 1-KLSS and Safe-1-KLSS try to solve a subgame S by solving the order-1 subgames, we would like to remark on the critical differences between them as follows:

Principle: 1-KLSS *converts* a common-knowledge subgame to an order-1 subgame, and reduces $M_{CK}^{\sigma^S}$ to $M_{KL}^{\sigma^S}$. So, 1-KLSS should only solve one S^I for a subgame S in principle, otherwise, it is unsafe. In contrast, Safe-1-KLSS *decomposes* S to many order-1 subgames and decomposes $M_{CK}^{\sigma^S}$ to $\{M_{SKL}^{\sigma^{S_I}}\}_{I \in \mathcal{I}_1^{S_{top}}}$. It solves S by solving (maximizing $M_{SKL}^{\sigma^{S_I}}$) each order-1 subgame independently.

Safety: 1-KLSS is unsafe, since maximizing $M_{KL}^{\sigma^S}$ in each order-1 subgame does not guarantee that $M_{CK}^{\sigma^S} \geq 0$. In contrast, Safe-1-KLSS is safe, since $M_{CK}^{\sigma^S}$ is lower bounded by the belief-weighted sum of $M_{SKL}^{\sigma^{S_I}}$ (Eq. 6).

Practice: 1-KLSS scales up the gift when maximizing $M_{KL}^{\sigma^S}$. This can be seen when we divide Eq. (5) by $\beta_{-2}^{\sigma}(I|I_2)$. Note that $\beta_{-2}^{\sigma}(I|I_2) \in [0, 1]$. In contrast, Safe-1-KLSS does not scale the gift and guarantees the total gift used by all the order-1 subgames does not exceed $R^{\sigma_2^S}$.

The rest of this section considers the efficiency of Safe-1-KLSS. Safe-1-KLSS has to guarantee $M_{SKL}^{\sigma^{S_I}}(I_2) \geq 0$ for every $I \in \mathcal{I}_1^{S_{top}}$ and every $I_2 \in \mathcal{I}_2^{S_{top}}$. This may be too conservative and inefficient since 1) some infosets of P_1 may not use the gift, which can be redistributed to the other infosets in $\mathcal{I}_1^{S_{top}}$; 2) some infosets of P_2 may not be reached by any best response so we do not need to concern the margins there to guarantee safety. We thereby propose two treatments, one for each point, to improve efficiency.

The first treatment is to scale up the gift. This allows some order-1 subgames to use more of the gift while assuming other subgames use less. Recall that the algorithm is still safe as long as the total gift used does not exceed $R_2^{\sigma^S}(I_2)$. In this paper, the gift is scaled up by a factor of two by default. Different scale factors are tested in Appendix C.

For the second point, we can weigh the infoset $I_2 \in \mathcal{I}_2^{S_{top}}$ according to the reaching probability of any best response of the resolved strategy σ_1'. The reason is that the best response value $u_1(\sigma_1', BR(\sigma_1'))$, which determines the exploitability, is a sum of payoffs weighted by the reaching probabilities of $BR(\sigma_1')$ ultimately. Therefore, instead of maximizing every margin for $I_2 \in \mathcal{I}_2^{S_{top}}$, it should be more effective in reducing the exploitability by maximizing the reaching-probability-weighted sum of them. While $BR(\sigma_1')$ and its reaching probability are not known before we solve σ_1', we propose to "guess" them online. One option is to assume the opponent will reach according to the blueprint strategy, just as in Unsafe subgame solving. To make it safer, we can allow the opponent to choose among a set of strategies. When the opponent can choose any strategy, P_1 is forced to guarantee the safety against any strategy. We will present the method, named Opponent-Limited Subgame Solving (OLSS), in the next section.

5. Opponent-Limited Subgame Solving

In this section, we propose OLSS-I, which limits how the opponent may reach the subgame, to make Safe-1-KLSS more efficient. Then, for even larger games that safe-1-KLSS and OLSS-I are incapable of, we propose OLSS-II to further reduce the size of the subgame, by limiting how the opponent may act in the subgame.

5.1. OLSS-I

OLSS-I allows the opponent to choose a mixture of a set of predefined strategies $\Sigma_2^N = \{\sigma_2^1, \ldots, \sigma_2^i, \ldots, \sigma_2^N\}$ to reach the subgame. This is partially inspired by depth-limited subgame solving (Brown et al., 2018), which allows the opponent to choose among a set of strategies for the remainder of the game at the depth limit. For any subgame S, define the *opponent-limited margin*:

$$\begin{aligned} M_{OL1}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) =& CBV_2^{\langle \sigma_1, \sigma_2^i \rangle}(S_{top}) \\ & - CBV_2^{\langle \sigma_1[S \leftarrow \sigma_1^S], \sigma_2^i \rangle}(S_{top}), \end{aligned} \tag{8}$$

OLSS-I is subject to $M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) \geq 0, \forall \sigma_2^i \in \Sigma_2^N$. In this way, we consider the subgame S as a whole and weigh each history h according to the belief $\beta^{\langle \cdot, \sigma_2^i \rangle}(h|S_{top})$. For the order-1 knowledge subgame S_I in Safe-1-KLSS, we only need to consider the histories $h \in I$, and maximize the margin: $\max_{\sigma_1^{S_I} \in \Sigma_1^{S_I}} \min_{\sigma_2^i \in \Sigma_2^N} M_{OLKL1}^{\langle \sigma_1^S, \sigma_2^i \rangle}(I)$, where

$$M_{OLKL1}^{\langle \sigma_1^S, \sigma_2^i \rangle}(I) = \min_{\sigma_2^{S_I} \in \Sigma_2^{S_I}} \left\{ d^{\langle \sigma_1^{S_I}, \sigma_2' \rangle}(I) + R^{\sigma_2'}(I) \right\}, \tag{9}$$

$$\begin{aligned} d^{\langle \sigma_1^{S_I}, \sigma_2' \rangle}(I) &= v_1^{\langle \sigma_1^{S_I}, \sigma_2' \rangle}(I) - v_1^{\langle \sigma_1, \sigma_2' \rangle}(I), \\ R^{\sigma_2'}(I) &= \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) R^{\sigma_2^{S_I}}(I_2(h)), \end{aligned}$$

and $\sigma_2' = \sigma_2^{i[S_I \leftarrow \sigma_2^{S_I}]}$. Clearly, if $\Sigma_2^N = \Sigma_2$ or Σ_2^N includes all the pure undominated P_2 strategies that can reach I, OLSS-I is safe:

Theorem 5.1. *Given a strategy σ_1, a common-knowledge subgame S, a set of subgames $\mathbb{S} = \{S_I | I \in \mathcal{I}_1^{S_{top}}\}$, and a strategy $\sigma_1^{S_I}$ for each subgame S_I produced by applying OLSS-I, let σ_1' be the strategy that plays according to $\sigma_1^{S_I}$ for each subgame S_I and σ_1 elsewhere. If Σ_2^N includes all the pure undominated P_2 strategies, then, $e(\sigma_1') \leq e(\sigma_1)$.*

We prove the theorem by showing $M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) \geq 0$, $\forall \sigma_2^i \in \Sigma_2^N$. The full proof is in Appendix A. Of course, including all the pure undominated strategies is impractical. In many cases, using the blueprint strategy may still produce safe strategies. Note that when $\Sigma_2^N = \{\sigma_2\}$, OLSS-I is equivalent to Unsafe solving, which performs well in many cases (Brown & Sandholm, 2019), except that OLSS-I is for order-1 subgames. To make the solved strategy safer, it is likely enough to include only a few "meta-strategies". In our experiments, OLSS-I can always produce low-exploitability strategies in poker games using the blueprint strategy and three "biased" strategies (Brown et al., 2018). Since the strategies are predefined, it allows us to use Monte Carlo CFR (MCCFR) (Lanctot et al., 2009) to solve the gadget game, which samples the root nodes according to the belief of the histories. Doing so also makes OLSS-I more efficient.

OLSS-I can also be applied for solving common-knowledge subgames and use Resolving or Maxmargin solving to find a strategy σ^S that satisfies $M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) \geq 0, \forall \sigma_2^i \in \Sigma_2^N$, which can be seen as an extension of Unsafe solving.

5.2. OLSS-II

Although Safe-1-KLSS and OLSS-I can solve much larger subgames than previous common-knowledge subgame solving, They still do not apply to games with large order-1 closures. For example, in two-player Mahjong, the size of an order-1 closure is about 10^{11}, making it impossible to solve the subgame online. For this kind of IIGs, we need a new method whose subgame size is independent of the size of any order knowledge closure. One idea is to limit how the opponent may act in the subgame. In this subsection, we propose **OLSS-II** to allow the opponent to choose a mixture of a set of predefined strategies for the *entire* game.

Specifically, given a blueprint σ and a strategy $\sigma_2^i \in \Sigma_2^N$, define the *opponent-limited margin* as

$$M_{OL2}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) = CBV_2^{\langle \sigma_1, \sigma_2^i \rangle}(S_{top}) - v_2^{\langle \sigma_{1[S \leftarrow \sigma_1^S]}, \sigma_2^i \rangle}(S_{top}). \tag{10}$$

For an order-1 subgame S_I in Safe-1-KLSS, the target is $\max_{\sigma_1^{S_I} \in \Sigma_1^{S_I}} \min_{\sigma_2^i \in \Sigma_2^N} M_{OLKL2}^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I)$, where

$$\begin{aligned} M_{OLKL2}^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I) =& d^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I) + R^{\sigma_2^i}(I), \\ d^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I) =& v_1^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I) - v_1^{\langle \sigma_1, \sigma_2^i \rangle}(I), \\ R^{\sigma_2^i}(I) =& \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) R^{\sigma_2^i}(I_2(h)). \end{aligned} \tag{11}$$

Clearly, OLSS-II can be seen as a special case of OLSS-I with $\sigma_2^{S_I} = \sigma_2^i$ in S_I. Therefore, if Σ_2^N includes all the pure undominated strategies, OLSS-II is safe.

More importantly, when σ_1 and σ_2^i are given, $v_1^{\langle \sigma_1, \sigma_2^i \rangle}(I)$, $R^{\sigma_2^i}(I)$, and $v_1^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I)$ are only functions of $\sigma_1^{S_I}$ as in a single-agent environment. Therefore, after P_2 has chosen its strategy for the entire game, we can treat P_2 as a part of the environment, and define an "environmental model" that takes actions for both P_2 and chance according to predefined distributions. As a result, the subgame tree can be reduced to the infoset tree whose nodes are infosets of P_1. Specifically, given a strategy σ, we construct a model E for infosets $I \in \mathcal{I}_1$, $P(I) = -1$:

$$E_{-1}^{\sigma}(I, a) = \frac{\sum_{h \in I} \pi_{-1}^{\sigma}(h) \sigma(h, a)}{\sum_{h \in I} \pi_{-1}^{\sigma}(h)}. \tag{12}$$

Also, the payoff at terminal infoset I of P_1 is replaced by

$$V_1^{\sigma}(I) = \frac{\sum_{z \in I} \pi_{-1}^{\sigma}(z) u(z)}{\sum_{z \in I} \pi_{-1}^{\sigma}(z)}. \tag{13}$$

Here the definition of infoset is extended to terminal nodes. So, the counterfactual value at infoset I of P_1 can be computed as in a single-agent environment:

$$v_1^{\sigma}(I) = \sum_{I \sqsubseteq I', I' \subseteq Z} \pi^{\langle \sigma_1, E_{-1}^{\sigma} \rangle}(I, I') V_1^{\sigma}(I'). \tag{14}$$

Then $v_1^{\langle \sigma_1, \sigma_2^i \rangle}(I)$, $R^{\sigma_2^i}(I)$, and $v_1^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I)$ can be computed accordingly. Obviously, traversing the infoset tree is much more efficient than traversing the original history tree. This makes OLSS-II the only subgame-solving algorithm suitable for large IIGs, e.g., two-player Mahjong. Although limiting the strategy of P_2 may result in a more exploitable strategy, we find that in two-player Mahjong, including only one or two P_2 strategies in Σ_2^N is enough to gain significant benefit. When $N = 1$, OLSS-II is reduced to a local best response algorithm against the one opponent model in Σ_2^N. Nevertheless, if we have a model that estimates the opponent's strategy well, the resolved strategy by OLSS-II may even be better than a Nash-equilibrium strategy.

6. Gadget Games and Examples

In this section, we present how to construct gadget games for Safe-1-KLSS and OLSS to use iterative algorithms, e.g., CFR, to resolve the subgames. Take the game shown in Figure 2 as an example and suppose P_1 has reached I_a and σ_1 is the uniform strategy, of which the exploitability is $e(\sigma_1) \approx 0.166$. The Safe-1-KLSS Maxmargin gadget game is shown on the left of Figure 3. The gadget game is constructed as follows: **1)** The order-1 subgame is first cloned, which is a set of subtrees rooted at $h \in I$. **2)** Upon I, P_2 is allowed to choose an infoset $I_2 \in \mathcal{I}_2^{S_{top}}$. **3)** $R^{\sigma^{S_I}}$ and $v_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}$ are precomputed and added to the payoffs for every $\sigma_2^{S_I} \in \Sigma_2^{S_I}$. In more complicated cases, another two passes may be needed for computing them. Note that the "environmental model" trick can also be used in Safe-1-KLSS and OLSS-I to compute the gifts and counterfactual values online as σ_1 is known. For Safe-1-KLSS, the solved strategy at I_a is the same as the blueprint. This is due to the artifact that P_2 can choose an infoset in Maxmargin solving. Note that P_2 does not act before I_a in the original game. OLSS-I can eliminate this artifact since it considers the opponent's strategy rather than its infosets.

In the middle of Figure 3, the OLSS-I gadget game is shown. The construction is similar to that in Safe-1-KLSS, except that P_2 chooses a strategy rather than an infoset. Since P_2 has not acted, it is safe to only include an arbitrary P_2 strategy in Σ_2^N. After solving the gadget game, the strategy at I_a increases the probability of choosing T to 0.6. This reduces the exploitability by about 0.066. When the subgame at I'_a is also solved independently, the total reduction of the exploitability is approximately 0.122. On the right of Figure 3 is the gadget game of OLSS-II. OLSS-II is not safe. In this example, the exploitability is increased by 0.066.

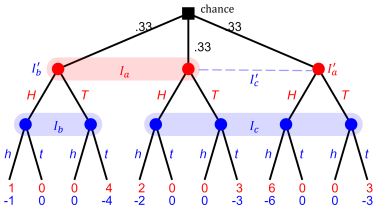


Figure 2: A game modified from 100-matching pennies.

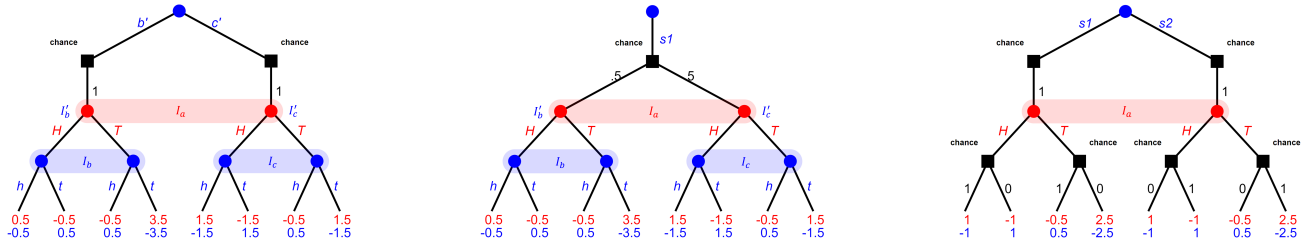


Figure 3: **Left**: The Safe-1-KLSS maxmargin gadget game constructed from I_a. **Middle**: The OLSS-I gadget game. P_2 can only choose action $s1$ at the beginning, which represents an arbitrary strategy. **Right**: The OLSS-II gadget game. P_2 can choose actions $s1$ or $s2$ at the root node to use strategy σ_2^1 or σ_2^2 in the subgame, respectively. P_2 nodes are replaced by chance nodes, as its strategy is fixed except at the root node.

7. Experimental Results

In this section, we test Safe-1-KLSS and two OLSS algorithms in multiple IIGs, including small and medium poker games, whose infoset sizes vary from 6 to 2600, and one large-scale two-player Mahjong game (Fu et al., 2022a), whose infoset size is approximately 10^{11}. Safe-1-KLSS and OLSS-I are evaluated in all the poker games. OLSS-II is only evaluated in Mahjong, not in poker, because it is designed for larger-scale games. Indeed, OLSS-II is the only algorithm available for the Mahjong benchmark.

7.1. Results in Poker Games

We first test our algorithms in four Leduc poker (Southey et al., 2012) and one Flop hold'em Poker (FHP) (Brown et al., 2019). In Table 2, Leduc(n_s, n_r, n_h) is a Leduc game that has n_s suits $\times$ n_r ranks and n_h hole cards. We use these games to test the scalability of the algorithms. FHP is a simplified two-player heads-up limit Texas hold'em. The description of the games can be found in Appendix B. The blueprint strategy in each game is generated by MCCFR (Lanctot et al., 2009), and Chance Sampling CFR (CSCFR) (Johanson et al., 2012) is used to solve the gadget games. Subgame solving is applied in each game immediately after the community card(s) are dealt. The solved strategies are combined with the blueprint to evaluate the exploitability.

As shown in Table 2, Safe-1-KLSS reduces the exploitability in every game, while 1-KLSS usually increases the exploitability. We can also see that Unsafe solving performs exceptionally well in large Leduc pokers and FHP. However, Unsafe solving increases the exploitability in Leduc(2,3,1). Besides, OLSS-I reduces the exploitability and is faster than Safe-1-KLSS in every game, and it achieves lower exploitability than Maxmargin solving in most of the Leduc games. So, opponent limiting is effective in poker games.

In Figure 4, the exploitability curve of each algorithm in FHP is shown. The X-axis is the number of iterations of CSCFR for solving the subgames. As we can see, Safe-1-KLSS and OLSS-I can reduce the exploitability after about 10^3 iterations, while 1-KLSS significantly increases the exploitability (from 10.640 to 97.313). Furthermore, Safe-1-KLSS and OLSS-I are about **100** times faster than the common-knowledge solving algorithms. This is because the size of the order-1 subgames they solve is only $1/1326$ of the common-knowledge subgame in FHP. In our experiments, the subgame size is further reduced using hand isomorphism algorithm (Waugh, 2013). In Figure 5, it is shown that as the game size increases, the number of iterations needed to reduce the exploitability by 20% increases more slowly in OLSS-I, since the subgame size in Safe-1-OLSS and OLSS-I is $O(\overline{I^1})$ rather than $O(\overline{I^\infty})$ in common-knowledge subgame solving. The results also show that OLSS-I is more efficient than Safe-1-KLSS. So, opponent limiting is effective and efficient in poker games.

7.2. Results in Two-Player Mahjong

In this section, we test OLSS-II in Two-player Mahjong. The blueprint strategy is trained using ACH algorithm (Fu et al., 2022a), which is an actor-critic algorithm for large-scale IIGs. For comparison, we also trained an agent using PPO algorithm (Schulman et al., 2017). The agents are referred to as ACH and PPO in the rest of the paper. As we have mentioned, using one or two opponent strategies significantly improves the strategy. For the Mahjong game, two environmental models are trained and used. The first one is a *Random (R) model* [1] that randomly discards a tile according to the distribution of the remaining tiles. The second one is the environmental model of ACH. We ignore $R^{\sigma^{SI}}(I)$ in this test since it is difficult to learn or compute in such a large-scale game. It is also less useful when N is small. We run our search in a depth-limited fashion similar to previous online searches in large-scale games (Silver et al., 2016; 2017; Brown et al., 2018; Zhang & Sandholm, 2021). The depth limit is set to eight steps in our

[1] We use depth-limited search, so the "random" model is not entirely random. Considering the highly stochastic nature of the game, we believe a random model in a depth-limited fashion is a good environmental model for the game.

Table 2: Exploitability of Subgame solving in poker games.

Game	Blueprint	Unsafe	Maxmargin	Resolving	1-KLSS	Safe-1-KLSS	OLSS-I
Leduc(2,3,1)	0.150	0.195	0.109	**0.089**	0.190	0.120	0.091
Leduc(2,3,1)	0.150	0.142	0.115	0.113	0.132	0.127	**0.099**
Leduc(2,13,1)	0.150	**0.044**	0.084	0.065	0.453	0.127	0.097
Leduc(2,13,2)	0.150	**0.077**	0.127	0.089	1.091	0.128	0.107
Leduc(2,13,3)	0.150	**0.089**	0.113	0.090	1.108	0.117	0.104
FHP	10.640	**5.077**	6.153	6.142	97.313	8.334	6.659

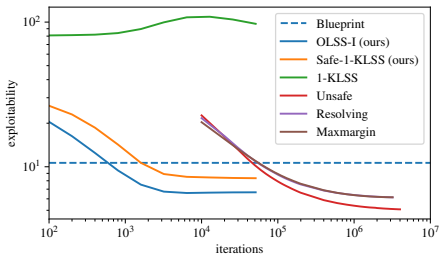


Figure 4: Exploitability of OLSS-I and common-knowledge subgame solving algorithms in FHP.

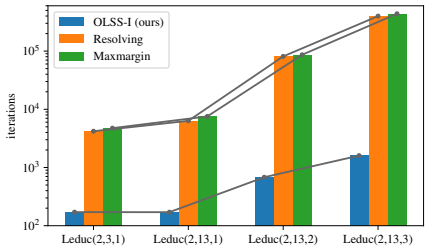


Figure 5: Number of iterations needed to reduce the exploitability by 20% in Leduc poker games.

Table 3: Head-to-head performance of ACH v.s. the opponents before and after applying OLSS-II. The values are given with 95% confidence interval. In the setting column, "R" and "ACH" indicate whether the Random model and the ACH model are used in the subgame solving. For comparison, a best response (estimated using a PPO agent) can win an ACH agent by no more than 0.75 (fan).

Opponent	Blueprint (fan)	Setting	OLSS-II (fan)
ACH	0.00 ± 0.00	R	0.14 ± 0.13
ACH	0.00 ± 0.00	ACH	0.18 ± 0.13
PPO	0.05 ± 0.13	R	0.22 ± 0.13
PPO	0.05 ± 0.13	ACH	0.20 ± 0.13
PPO	0.05 ± 0.13	R + ACH	0.22 ± 0.13

experiments. At the depth limit, we return the value of the blueprint. More details about the blueprint strategy and the environmental models are provided in Appendix D.

As shown in Table 3, OLSS-II significantly improves the performance of ACH regardless of whether the opponent is ACH itself or PPO. Among the settings, OLSS-II with the Random environmental model is doing surprisingly well. So, although the environmental model does not reflect any opponent's strategy, it captures the dynamics of the game and helps the agent refine its strategy. Besides, when the opponent is ACH itself, OLSS-II performs better if the ACH environmental model is used. This suggests that OLSS-II can be combined with opponent-modeling methods (Ganzfried & Sandholm, 2015; Fu et al., 2022b).

8. Conclusions and Future Work

For large-scale IIGs, we proposed Safe-1-KLSS and two types of OLSS. Specifically, we revisited knowledge-limited subgame solving and proposed Safe-1-KLSS, which is proven safe when applied to every infoset during play. Based on Safe-1-KLSS, a more efficient OLSS-I, which limits how the opponent can reach the subgame, is proposed. For larger IIGs, we propose OLSS-II, which only allows the opponent to choose among a set of predefined strategies for the entire game. The experiments in poker games show that Safe-1-KLSS and OLSS-I are effective and efficient, by orders of magnitude faster than previous common-knowledge subgame solving. Despite lacking theoretical guarantees, OLSS-II is the only algorithm applicable for two-player Mahjong, and it significantly improves the performance.

With Safe-1-KLSS and OLSS-I, online subgame solving can be applied to larger IIGs that were impossible with previous common-knowledge subgame solving. They can also be combined with previous techniques, such as Reach-Maxmargin and depth-limited subgame solving, for more efficiency. For OLSS-II, combining it with opponent modeling is a promising direction. Finally, although using a small number of strategies in OLSS-I and OLSS-II is enough in our experiments, there are still some open questions: 1) how many strategies are needed to guarantee safety? 2) how do we find the smallest set of strategies for good performance?

References

Brown, N. and Sandholm, T. Safe and nested subgame solving for imperfect-information games. In *Advances in Neural Information Processing Systems*, pp. 689–699, 2017.

Brown, N. and Sandholm, T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Brown, N. and Sandholm, T. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019.

Brown, N., Sandholm, T., and Amos, B. Depth-limited solving for imperfect-information games. In *Advances in Neural Information Processing Systems 31*, pp. 7663–7674, 2018.

Brown, N., Lerer, A., Gross, S., and Sandholm, T. Deep counterfactual regret minimization. In *International Conference on Machine Learning*, volume 97, pp. 793–802, 2019.

Burch, N., Johanson, M., and Bowling, M. Solving imperfect information games using decomposition. In *AAAI Conference on Artificial Intelligence*, pp. 602–608, 2014.

Fu, H., Liu, W., Wu, S., Wang, Y., Yang, T., Li, K., Xing, J., Li, B., Ma, B., Fu, Q., and Yang, W. Actor-critic policy optimization in a large-scale imperfect-information game. In *International Conference on Learning Representations*, 2022a.

Fu, H., Tian, Y., Yu, H., Liu, W., Wu, S., Xiong, J., Wen, Y., Li, K., Xing, J., Fu, Q., and Yang, W. Greedy when sure and conservative when uncertain about the opponents. In *International Conference on Machine Learning,*, volume 162 of *Proceedings of Machine Learning Research*, pp. 6829–6848, 2022b.

Ganzfried, S. and Sandholm, T. Safe opponent exploitation. *ACM Transactions on Economics and Computation*, 3(2): 8:1–8:28, 2015. doi: 10.1145/2716322.

Gilpin, A. and Sandholm, T. A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation. In *National Conference on Artificial Intelligence and*, pp. 1007–1013, 2006.

Gilpin, A. and Sandholm, T. Lossless abstraction of imperfect information games. *Journal of the ACM*, 54(5):25, 2007.

Gilpin, A., Sandholm, T., and Sørensen, T. B. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *AAAI Conference on Artificial Intelligence*, pp. 50–57, 2007.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Johanson, M., Bard, N., Lanctot, M., Gibson, R. G., and Bowling, M. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 837–846, 2012.

Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. H. Monte Carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems*, pp. 1078–1086, 2009.

Lanctot, M., Lockhart, E., Lespiau, J., Zambaldi, V. F., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., Vylder, B. D., Saeta, B., Bradbury, J., Ding, D., Borgeaud, S., Lai, M., Schrittwieser, J., Anthony, T. W., Hughes, E., Danihelka, I., and Ryan-Davis, J. Openspiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019. URL http://arxiv.org/abs/1908.09453.

Moravcik, M., Schmid, M., Ha, K., Hladík, M., and Gaukrodger, S. J. Refining subgames in large imperfect information games. In *AAAI Conference on Artificial Intelligence*, pp. 572–578, 2016.

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Perolat, J., Vylder, B. D., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., McAleer, S., Elie, R., Cen, S. H., Wang, Z., Gruslys, A., Malysheva, A., Khan, M., Ozair, S., Timbers, F., Pohlen, T., Eccles, T., Rowland, M., Lanctot, M., Lespiau, J.-B., Piot, B., Omidshafiei, S., Lockhart, E., Sifre, L., Beauguerlange, N., Munos, R., Silver, D., Singh, S., Hassabis, D., and Tuyls, K. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022. doi: 10.1126/science.add4679.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I.,

Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi: 10.1038/Nature24270.

Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., and Rayner, D. C. Bayes' bluff: Opponent modelling in poker. *CoRR*, abs/1207.1411, 2012. URL http://arxiv.org/abs/1207.1411.

Tammelin, O., Burch, N., Johanson, M., and Bowling, M. Solving heads-up limit texas hold'em. In *International Joint Conferences on Artificial Intelligence*, pp. 645–652, 2015.

Waugh, K. A fast and optimal hand isomorphism algorithm. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

Zhang, B. H. and Sandholm, T. Subgame solving without common knowledge. In *Advances in Neural Information Processing Systems*, pp. 23993–24004, 2021.

Zinkevich, M., Johanson, M., Bowling, M. H., and Piccione, C. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2007.

A. Proofs

A.1. Proof for Equation 3

Proof. The new objective function $d^{\sigma^S}(I_2)$ can be decomposed as follows:

$$\begin{aligned} d^{\sigma^S}(I_2) =& v_2^{\langle \sigma_1, \sigma_2^S \rangle}(I_2) - v_2^{\langle \sigma_{1[S \leftarrow \sigma_1^S]}, \sigma_2^S \rangle}(I_2) \\ =& \sum_{h \in I_2} \beta_{-2}^{\sigma}(h|I_2) \left((u_2^{\langle \sigma_1, \sigma_2^S \rangle}(h) - u_2^{\langle \sigma_1^S, \sigma_2^S \rangle}(h) \right) \\ =& \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I \cap I_2} \beta_{-2}^{\sigma}(h|I_2) \right) \sum_{h \in I \cap I_2} \frac{\beta_{-2}^{\sigma}(h|I_2)}{\sum_{h \in I \cap I_2} \beta_{-2}^{\sigma}(h|I_2)} \left(u_2^{\langle \sigma_1, \sigma_2^S \rangle}(h) - u_2^{\langle \sigma_1^S, \sigma_2^S \rangle}(h) \right) \\ =& \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I \cap I_2} \beta_{-2}^{\sigma}(h|I_2) \right) d^{\sigma^{S_I}}(I_2) \\ =& \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) d^{\sigma^{S_I}}(I_2). \end{aligned} \tag{15}$$

The second equality is according to the definition of counterfactual value. In the last line, we use $\beta_{-2}^{\sigma}(I|I_2)$ to denote $\sum_{h \in I \cap I_2} \beta_{-2}^{\sigma}(h|I_2)$. Note that $\sum_{h \in I \cap I_2} \beta_{-2}^{\sigma}(h|I_2) \geq 0$ and $\sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) = 1$.

□

A.2. Proof for Equation 6

Proof. For a common-knowledge subgame S, the common-knowledge margin is:

$$M_{CK}^{\sigma^S}(I_2) = \min_{\sigma_2^S \in \Sigma_2^S} \left\{ d^{\sigma^S}(I_2) + R^{\sigma_2^S}(I_2) \right\}. \tag{16}$$

Assume $d^{\sigma^S}(I_2) + R^{\sigma_2^S}(I_2) \geq 0$ for every $I_2 \in \mathcal{I}_2^{S_{top}}$. Note that σ_1^S must exist such that $d^{\sigma^S}(I_2) + R^{\sigma_2^S}(I_2) \geq 0$ since it is true when σ_1^S equals σ_1 in the subgame.

For $M_{CK}^{\sigma^S}(I_2)$, we have

$$\begin{aligned} M_{CK}^{\sigma^S}(I_2) =& \min_{\sigma_2^S \in \Sigma_2^S} \left\{ d^{\sigma^S}(I_2) + R^{\sigma_2^S}(I_2) \right\} \\ =& \min_{\sigma_2^S \in \Sigma_2^S} \left\{ \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^S}(I_2) \right\} \\ =& \min_{\sigma_2^S \in \Sigma_2^S} \left\{ \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) \left(d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^S}(I_2) \right) \right\} \\ \geq& \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) \min_{\sigma_2^S \in \Sigma_2^S} \left\{ d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^S}(I_2) \right\} \\ =& \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) \min_{\sigma_2^{S_I} \in \Sigma_2^{S_I}} \left\{ d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^{S_I}}(I_2) \right\}. \end{aligned} \tag{17}$$

The second equality is because of $\beta_{-2}^{\sigma}(I|I_2) \geq 0$ and $\sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) = 1$. The inequality is according to Jensen's inequality. The last equality is because P_2 only needs to consider its strategy in S_I to minimize $d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^S}(I_2)$. More

specifically, since P_1 only changes its strategy in S_I, the counterfactual best response strategy of P_2 in the infosets that do not intersect with S_I is fixed and therefore does not affect $R^{\sigma_2^S}(I_2)$. So, $R^{\sigma_2^S}(I_2) = R^{\sigma_2^{S_I}}(I_2)$.

Therefore, we have

$$\begin{aligned} M_{CK}^{\sigma^S}(I_2) &\geq \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) \min_{\sigma_2^{S_I} \in \Sigma_2^{S_I}} \left\{ d^{\sigma^{S_I}}(I_2) + R^{\sigma_2^{S_I}}(I_2) \right\} \\ &= \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) M_{SKL}^{\sigma^{S_I}}(I_2). \end{aligned} \tag{18}$$

□

A.3. Proof for Theorem 4.1

Proof. According to the definition of $M_{CK}^{\sigma^S}$, we have

$$CBV^{\sigma_1[S \leftarrow \sigma_1^S]}(I_2) \leq CBV^{\sigma_1}(I_2) - \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) M_{SKL}^{\sigma^{S_I}}(I_2), \forall I_2 \in \mathcal{I}_2^{S_{top}}. \tag{19}$$

Then, according to Theorem 1 in (Moravcik et al., 2016), if $\pi_2^{BR(\sigma')}(I_2^*) > 0$ for some $I_2^* \in \mathcal{I}_2^{S_{top}}$, we have

$$e(\sigma_1') \leq e(\sigma_1) - \sum_{h \in I_2^*} \pi_{-2}^{\sigma_1}(h) \sum_{I \in \mathcal{I}_1^{S_{top}}} \beta_{-2}^{\sigma}(I|I_2) M_{SKL}^{\sigma^{S_I}}(I_2^*). \tag{20}$$

□

A.4. Proof for Theorem 5.1

Proof. We first prove that for any $\sigma_2^i \in \Sigma_2^N$,

$$M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) \geq 0. \tag{21}$$

As we have defined before, the opponent-limited margin is

$$M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) = \min_{\sigma_2^S \in \Sigma_2^S} \left\{ CBV_2^{\langle \sigma_1, \sigma_2^i \rangle}(S_{top}) - v_2^{\langle \sigma_1[S \leftarrow \sigma_1^S], \sigma_2^i[S \leftarrow \sigma_2^S] \rangle}(S_{top}) \right\}. \tag{22}$$

For $CBV_2^{\langle \sigma_1, \sigma_2^i \rangle}(S_{top})$, we have

$$\begin{aligned} CBV_2^{\langle \sigma_1, \sigma_2^i \rangle}(S_{top}) &= \sum_{h \in S_{top}} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) CBV^{\sigma_1}(I_2(h)) \\ &= \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) \right) \sum_{h \in I} \frac{\beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top})}{\sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top})} CBV^{\sigma_1}(I_2(h)) \\ &= \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) \right) \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) CBV^{\sigma_1}(I_2(h)). \end{aligned} \tag{23}$$

Note that $\beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) / \sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) = \beta_{-1}^{\sigma_2^i}(h|I)$ since P_1 has the same reaching probability for each $h \in I$. Similarly, for $v_2^{\langle \sigma_1[S \leftarrow \sigma_1^S], \sigma_2^i[S \leftarrow \sigma_2^S] \rangle}(S_{top})$, we have

$$v_2^{\langle \sigma_1[S \leftarrow \sigma_1^S], \sigma_2^i[S \leftarrow \sigma_2^S] \rangle}(S_{top}) = \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) \right) \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) u_2^{\langle \sigma_1^S, \sigma_2^S \rangle}(h), \tag{24}$$

and

$$\begin{aligned} v_2^{\langle \sigma_1, \sigma_2^i[S \leftarrow \sigma_2^S] \rangle}(S_{top}) &= \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) \right) \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) u_2^{\langle \sigma_1, \sigma_2^S \rangle}(h) \\ &= \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) \right) \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) v_2^{\langle \sigma_1, \sigma_2^S \rangle}(I_2(h)). \end{aligned} \tag{25}$$

So, the opponent-limited margin is

$$\begin{aligned} & M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) \\ &= \min_{\sigma_2^S \in \Sigma_2^S} \left\{ CBV_2^{\langle \sigma_1, \sigma_2^i \rangle}(S_{top}) - v_2^{\langle \sigma_1[S \leftarrow \sigma_1^S], \sigma_2^i[S \leftarrow \sigma_2^S] \rangle}(S_{top}) \right\} \\ &= \min_{\sigma_2^S \in \Sigma_2^S} \sum_{I \in \mathcal{I}_1^{S_{top}}} \left(\sum_{h \in I} \beta^{\langle \sigma_1, \sigma_2^i \rangle}(h|S_{top}) \right) \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(u_2^{\langle \sigma_1, \sigma_2^S \rangle}(h) - u_2^{\langle \sigma_1^S, \sigma_2^S \rangle}(h) + CBV^{\sigma_1}(I_2(h)) - v_2^{\langle \sigma_1, \sigma_2^S \rangle}(I_2(h)) \right) \\ &\geq \min_{I \in \mathcal{I}_1^{S_{top}}} \min_{\sigma_2^S \in \Sigma_2^S} \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(u_2^{\langle \sigma_1, \sigma_2^S \rangle}(h) - u_2^{\langle \sigma_1^S, \sigma_2^S \rangle}(h) + CBV^{\sigma_1}(I_2(h)) - v_2^{\langle \sigma_1, \sigma_2^S \rangle}(I_2(h)) \right) \\ &= \min_{I \in \mathcal{I}_1^{S_{top}}} \min_{\sigma_2^{S_I} \in \Sigma_2^{S_I}} \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(u_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(h) - u_2^{\langle \sigma_1^{S_I}, \sigma_2^{S_I} \rangle}(h) + CBV^{\sigma_1}(I_2(h)) - v_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(I_2(h)) \right). \end{aligned} \tag{26}$$

For $\sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(u_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(h) - u_2^{\langle \sigma_1^{S_I}, \sigma_2^{S_I} \rangle}(h) \right)$, we know that

$$\begin{aligned} & \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(u_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(h) - u_2^{\langle \sigma_1^{S_I}, \sigma_2^{S_I} \rangle}(h) \right) \\ =& \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(u_1^{\langle \sigma_1^{S_I}, \sigma_2^{S_I} \rangle}(h) - u_1^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(h) \right) \\ =& v_1^{\langle \sigma_1^{S_I}, \sigma_2^i[S_I \leftarrow \sigma_2^{S_I}] \rangle}(I) - v_1^{\langle \sigma_1, \sigma_2^i[S_I \leftarrow \sigma_2^{S_I}] \rangle}(I) \\ =& d^{\langle \sigma_1^{S_I}, \sigma_2^i[S_I \leftarrow \sigma_2^{S_I}] \rangle}(I). \end{aligned} \tag{27}$$

Besides, for $\sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(CBV^{\sigma_1}(I_2(h)) - v_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(I_2(h)) \right)$, we know

$$\begin{aligned} & \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) \left(CBV^{\sigma_1}(I_2(h)) - v_2^{\langle \sigma_1, \sigma_2^{S_I} \rangle}(I_2(h)) \right) \\ =& \sum_{h \in I} \beta_{-1}^{\sigma_2^i}(h|I) R^{\sigma_2^{S_I}}(I_2(h)) \\ =& R^{\sigma_2^i[S_I \leftarrow \sigma_2^{S_I}]}(I). \end{aligned} \tag{28}$$

Combining the above equations, we get

$$\begin{aligned} & M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) \\ \geq & \min_{I \in \mathcal{I}_1^{S_{top}}} \min_{\sigma_2^{S_I} \in \Sigma_2^{S_I}} \left(d^{\langle \sigma_1^{S_I}, \sigma_2^i[S_I \leftarrow \sigma_2^{S_I}] \rangle}(I) + R^{\sigma_2^i[S_I \leftarrow \sigma_2^{S_I}]}(I) \right) \\ = & \min_{I \in \mathcal{I}_1^{S_{top}}} M_{OLKL1}^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I). \end{aligned} \tag{29}$$

Since OLSS-I guarantees that $M_{OLKL1}^{\langle \sigma_1^{S_I}, \sigma_2^i \rangle}(I) \geq 0$ for any $I \in \mathcal{I}_1^{S_{top}}$ and any strategy in Σ_2^N, $M_{OL}^{\langle \sigma_1^S, \sigma_2^i \rangle}(S_{top}) \geq 0$. When Σ_2^N includes all the pure non-dominated strategies, it means that any best response, which is a mixed strategy of some pure

non-dominated strategies, in the entire game can not get more payoff from S. So, we have $u_2(\sigma_{1[S\leftarrow\sigma_1^S]}, BR(\sigma_{1[S\leftarrow\sigma_1^S]})) \leq u_2(\sigma_1, BR(\sigma_1))$ and $e(\sigma_{1[S\leftarrow\sigma_1^S]}) \leq e(\sigma_1)$. □

B. Description of Benchmark Games

- **Leduc(2,3,1)** (Southey et al., 2012) is a two-player zero-sum EFG. It can be considered as a simplified Heads-up Limit Texas hold'em (HULH).[2] In Leduc, there are two suits of cards, each suit comprises three ranks, and two rounds of betting are allowed. At the beginning of the game, each player places an ante of one chip in the pot and is dealt with one card, which is only visible to itself. In the first round of betting, player 1 has to choose an action between *Call* and *Raise*. Taking the action Call means that the player will place or has placed the same chips as the opponent and leaves the choice to the opponent. Taking the action Raise means that the player will place more chips than the opponent to the pot. It is only two raises allowed in the first round of betting. Sometimes when a player bets fewer chips than his opponent and is asked to take an action, he can choose to *Fold*. If he does so, the game is over, and the player loses all chips. The first round ends if one of the players has chosen Fold or if both players agree to end. If no player folds, a public card is revealed to both of the players and then the second round of betting takes place, with the same dynamic as the first round. After the two rounds of betting, if one of the players has a pair with the public card, that player wins the pot. Otherwise, the player with a higher private card wins. In the first round, the player taking action Raise should place 2 (named raise size) more chips to the pot than the opponent. In the second round, the raise size is 4. The infoset size and the common-knowledge closure size after dealing the community card are 4 and 20, respectively.

- **Leduc(2,13,1)** has the same rules as Leduc, except that it has two suits of thirteen cards. The infoset size and the common-knowledge closure size after dealing the community card are 24 and 600, respectively.

- **Leduc(2,13,2)** has the same rules as Leduc, except that it has two suits of thirteen cards and each player is dealt two cards rather than one. The infoset size and the common-knowledge closure size after dealing the community card are 253 and 75900, respectively.

- **Leduc(2,13,3)** has the same rules as Leduc, except that it has two suits of thirteen cards and each player is dealt three cards rather than one. The infoset size and the common-knowledge closure size after dealing the community card are 1540 and 3542000, respectively.

- **FHP** (Brown et al., 2019) is a simplified HULH that uses standard four suits of thirteen cards. At the beginning of the game, each player places an ante of 50 chips in the pot and is dealt two cards. It has the same dynamic in each round of betting as Leduc. However, it allows three raises and a raise size of 100 in each round of betting, and the first player to act in the second round is player 2. After the first round of betting, three public cards are revealed to the players. After two rounds of betting, the rank of the five cards (two private cards + three public cards) of each player is evaluated, and the player with the higher rank wins the pot. We use the standard hand evaluating method[3] used in HULH. The infoset size and the common-knowledge closure size after dealing the community cards are 1081 and 1271256, respectively.

- **Two-player Mahjong** (Fu et al., 2022a) is a simplified Mahjong game. The game rules are similar to Competition Mahjong. The corresponding game, "two-player Mahjong Master", is played by humans in Tencent mobile games (https://majiang.qq.com). The infoset size is about 10^{11}, and the common-knowledge closure size is about 10^{22}.

C. Additional Experimental Results

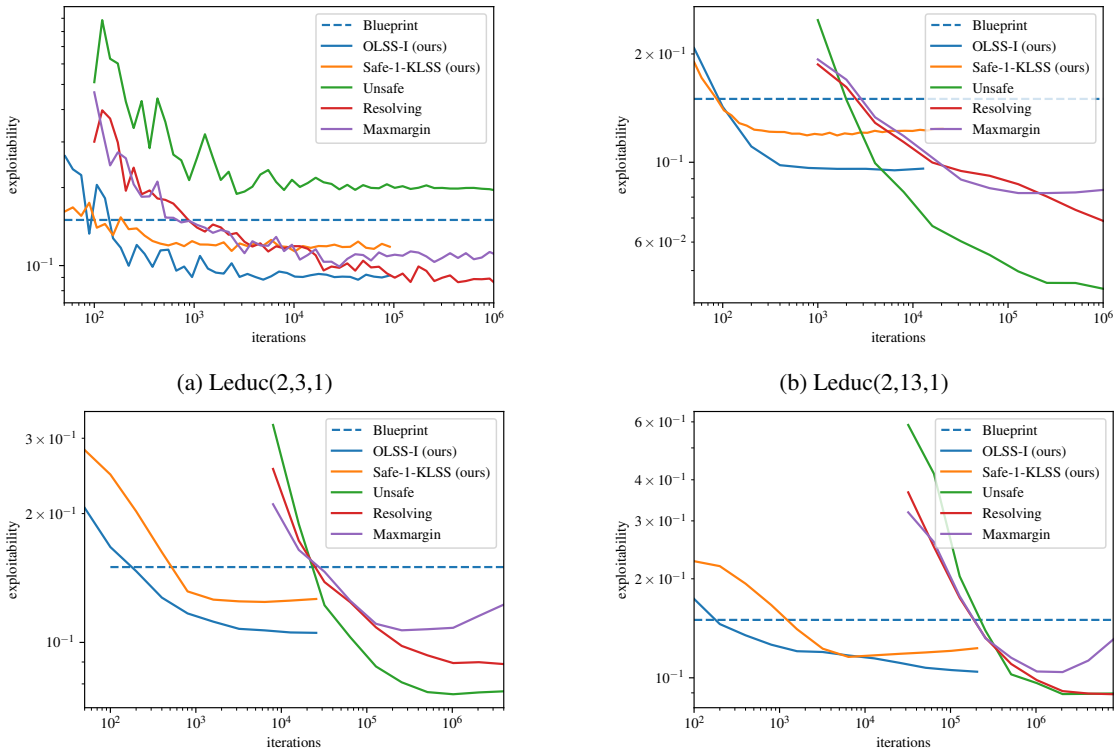
This section presents more results of Safe-1-KLSS and OLSS in poker games. The experiments are conducted based on the OpenSpiel project (Lanctot et al., 2019). The license is Apache-2.0.

C.1. Exploitability Curves

In Figure 6, the exploitability curves of Safe-1-KLSS, OLSS-I, and other algorithms in Leduc(2,3,1) poker games are given. The curve of 1-KLSS is not plotted as its values are much larger. As we have stated before, Unsafe increases the exploitability in Leduc poker but converges faster and better than the other algorithms in Leduc(2,13,1) and Leduc(2,13,2).

[2] https://en.wikipedia.org/wiki/Texas_hold_%27em

[3] https://en.wikipedia.org/wiki/List_of_poker_hands



(a) Leduc(2,3,1)

(b) Leduc(2,13,1)

(c) Leduc(2,13,2)

(d) Leduc(2,13,3)

Figure 6: Exploitability of the algorithms in Leduc poker games.

Also, as the game size increases, the proposed Safe-1-KLSS and OLSS-I are more efficient than other common-knowledge subgame solving.

C.2. Parameters in OLSS-I

OLSS-I assumes the opponent can choose among a set of N strategies Σ_2^N to reach the subgame. As discussed in the paper, using the blueprint and three "biased" strategies (Brown et al., 2018) is enough in Poker games to achieve good results. In this paper, the first biased strategy is biased to action *Fold*: the probability of action *Fold* is scaled up by five and then normalized. The other two biased strategies are biased to action *Call* and *Raise* by scaling up the probabilities, respectively.

In this subsection, we test OLSS-I with different numbers of opponent strategies in FHP, as shown in Figure 7. When $N = 1$, it assumes the opponent follows the blueprint to reach the subgame. When $N = 7$, another three biased strategies are included, which scale up the probabilities by 20. The results show that it is sufficient to limit the number of opponent strategies for reaching the subgame in FHP.

On the right of Figure 7, we test OLSS-I with different regret scale factors. As we can see, scaling up the regrets indeed improves performance. However, it can also hurt the performance when the scale factor is too large.

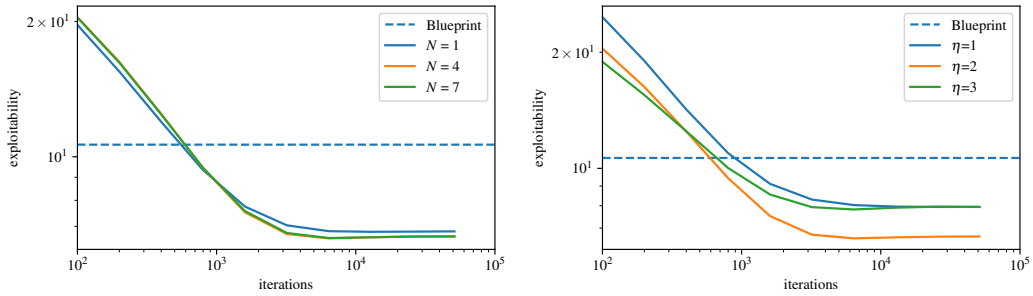


Figure 7: **Left**: Exploitability of OLSS-I with different numbers (denoted by N in the figure) of opponent strategies. **Right**: Exploitability of OLSS-I with different regret scale factors (denoted by η in the figure).

D. Mahjong

We now describe the details of experiments in two-player Mahjong. Two-player Mahjong is similar to Competition Mahjong. The difference is that in two-player Mahjong, there are only 72 tiles in total(Characters, Winds, Dragons, Flowers, Seasons). At the beginning of the game, each player is dealt 13 tiles, and each player takes action in turn. If the opponent just discarded, a player can choose Chow, Pong, or Kong the tile. Otherwise, he can draw a tile from the deck and discards one.

D.1. Environmental Model and Blueprint

For the environmental model, We need to calculate all the possible legal actions according to the remaining tiles in the current infoset. We use cross-entropy loss to train our environmental model. The final output is a mixed distribution of strategies that the opponent holds different tiles in hand. In environmental model training, the learning rate is 1e-3, and the batch size is 8192.

We use the same network architecture to train the blueprint and the environmental models, which is the same as ACH(Fu et al., 2022a). We use three residual blocks(He et al., 2016) layers and two fully-connected layers in our model. In the blueprint model, two branches output action probabilities and the infoset value. In the environmental model, only the action probabilities are predicted.

The Adam optimizer is used in our two-player Mahjong experiments. Blueprint models are trained for two days with 8 V100 GPUs and 1200 CPUs. And we use 8 V100 GPUs and 2400 CPUs to train the environmental model. Table 4 shows the overview of blueprint model hyper-parameters.

Parameter	Range	Best
Shared		
Ratio clip (ϵ)	-	0.5
GAE (λ)	-	0.95
Learning rate	{2.5e-3, 2.5e-4}	2.5e-4
Discount factor (γ)	-	0.995
Value loss coefficient (α)	-	0.5
Batch size	{4096, 8192}	8192
ACH		
Entropy coefficient (β)	{0.1, 1e-4}	5e-4
Logit threshold (l^{th})	-	8
PPO		
Entropy coefficient (β)	{0.1, 1e-4}	0.01

Table 4: Hyper-parameters used for the blueprint.

Parameter	value
Evaluation time	1000
Rollout steps	8
Exploration coefficient (c)	20
Value discount factor (γ)	1

Table 5: Hyper-parameters used in online search.

D.2. Online Search

The search in Mahjong is repeated for several simulations, which in our experiments is set to 1000. In each simulation, the search starts from a root infoset I_0 and finishes at the end of the game. The value at the root infoset is accumulated without discount($\gamma = 1$). At root infoset I_0, we select action according to pUCT algorithm, and at the rest of other infosets, actions are sampled according to their original strategies. In the part where the opponent chooses his strategy, the same pUCT algorithm is used.

We use pUCT in Mahjong because we found that if we use a CFR algorithm like experiments in poker, we need many simulations to make the average policy converge, which is time-consuming. Compared to the CFR algorithm, pUCT is a much faster algorithm. It considers prior probability and samples a trajectory instead of traversing the whole game tree. In our experiments, we can get a significant result by pUCT algorithm with 1000 simulation times, but the CFR algorithm with 5000 simulations is far from enough.

$$a^t = \underset{a \in A(I)}{\operatorname{argmax}} \left\{ Q(I,a) + \sigma(I,a) \cdot \frac{\sqrt{\sum_b N(I,b)}}{1 + N(I,a)} \cdot c \right\} \tag{30}$$

In (30), $Q(I,a)$ denotes the accumulated value from previous simulations, $P(I,a)$ is the probability of action a on infoset I, $N(I,a)$ is the number of visit times, and c is a hyper-parameter that controls the degree of exploration. Two-player Mahjong is a game with high variance, so a large parameter value $c = 20$ is preferred to encourage exploration and reduce the influence of $Q(I,a)$ in our experiments.

Due to the large number of infosets in two-player Mahjong, we simplified our experiments. During simulations, all the players are only allowed to take Discard-type actions. Instead of rollout to the end, each simulation ends after eight steps and returns a value from a value function. The value function is trained with the blueprint policy simultaneously.

D.3. Evaluation

Because of the high variance of two-player Mahjong, we need a large number of games to evaluate. In order to make our experimental results significant, we conduct 200000 games in our experiments. We randomly generated 100000 decks and

allowed players to swap their positions alternately in the same deck, which can help reduce the variance of evaluation.