Fast Algorithms for Poker Require Modelling it as a Sequential Bayesian Game

Vojtěch Kovařík[1*], **David Milec**[1†], **Michal Šustr**[1,2] and **Dominik Seitz**[1], **Viliam Lisý**[1]
[1]Artificial Intelligence Center, Faculty of Electrical Engineering,
Czech Technical University in Prague
vojta.kovarik@gmail.com, milecdav@fel.cvut.cz, {michal.sustr, dominik.seitz} @aic.fel.cvut.cz}
viliam.lisy@agents.fel.cvut.cz

Abstract

Many recent results in imperfect information games were only formulated for, or evaluated on, poker-like games. We argue that sequential Bayesian games constitute a natural class of games for generalizing these results. This model allows for an elegant formulation of the counterfactual regret minimization algorithm (CFR), called public-state CFR, which naturally lends itself to an efficient implementation. Empirically, solving a poker subgame with 10^7 states by public-state CFR takes 3 minutes and 700 MB while a comparable version of vanilla CFR takes 5.5 hours and 20 GB. Moreover, this formulation allows for exploiting domain-specific assumptions, leading to a *quadratic reduction* in asymptotic complexity (and a further empirical speedup) over vanilla CFR in poker and other domains. These results suggest the ability to represent poker as a Bayesian extensive game played a key role in the success of CFR-based methods. Finally, we extend public-state CFR to general extensive-form games, arguing that this extension enjoys some (but not all) benefits of the version for sequential Bayesian games.

1 Introduction

Poker has been a challenge for artificial intelligence research since the field's inception [16]. Recent progress has led to essentially solving the two-player limit variant with 10^{14} decision points [23] and outperforming professional human players not only in the two-player no-limit variant with 10^{170} decision-points but also in the analogous six-player game [4]. Research papers commonly evaluate novel algorithmic ideas on abstracted poker "river subgames" (10^7 histories), which can be quickly solved even in complete tabular representations.

Generalising these results to other games, however, proved difficult. On the conceptual level, the algorithms have been generalised to the full class of imperfect-information extensive-form games (EFGs)[1] [22, 21]. However, implementations of these algorithms often did not show similarly impressive scalability as in poker — algorithms for solving *general* EFGs are rarely evaluated on games with more than 10^7 histories.

We argue that this is due to the particular structure of poker, which is not shared by most other imperfect-information games. Indeed, the hidden cards are dealt to the players at the beginning of the game, and all subsequent chance events (public cards) and actions (bets) are publicly observable, which has two implications: First, the space needed to represent the hidden information remains constant and relatively small throughout the game. Second, inferring the opponent's possible hidden information is trivial. Furthermore, whether an action is legal never depends on hidden information. These simplifications do not hold even in most card games (where players can only play cards from their hand), not to mention other games such as imperfect-information variants of chess or computer games (where many of the opponent's actions go unobserved).

Some previous works explicitly discuss and exploit the properties of poker [9, 14, 10]. Similarly, the recent paper [20] acknowledges its use of a very specific version of the counterfactual regret minimization algorithm (CFR), though it does not go into details regarding the differences between this version and the vanilla CFR [27]. However, a vast majority of recent papers present their analysis for EFGs but only implement (and evaluate) their ideas on poker or games with near-identical structure (such as liar's dice). This is the case for example for [27, 5, 15, 3]. (A few notable exceptions include [7, 18, 12].) Importantly, these algorithms often employ a range of non-generalisable domain-specific tricks and speed-ups. We argue that these implementations are often optimised for poker to such a degree that they implement a different algorithm from the one described by the theory.

We aim to bridge this divide between theory and implementation for EFGs. We argue that the poker-specific ideas become natural and generalisable if we stop modelling it as a *general* EFG and instead view it as a

*Contact Author
†The second and third author contributed equally.

[1]EFG will, by default, refer to an *imperfect* inform. game.

Bayesian extensive game (BEG) with perfect (but incomplete) information. BEGs form a subclass of EFGs. They can be described [17] as extensive-form games in which (1) the players are each assigned a *type*. The joint probability distribution over types is common knowledge, but each player only knows their own type. (2) The players play a perfect information extensive-form game. (3) Rewards in this game depend not only on actions taken but also on the private types of all players.

We present a variant of counterfactual regret minimization for BEGs and argue that this is the algorithm typically used in existing poker literature. We show that running CFR on BEGs can be asymptotically more efficient on particular games with sparse or largely independent type distributions, such as poker. Our experiments show that while an implementation of CFR that runs on the standard EFG representation takes 5.5 hours to solve a river sub-game of poker, an implementation that runs on the BEG representation solves the same game in 1.6 minutes, i.e., approximately 200x faster. In particular, the results suggest that games that can be represented as BEGs can be solved by CFR substantially faster than more generic imperfect-information games that do not allow such representation. Consequently, we cannot expect the tabular game-solving algorithms for generic EFGs to be evaluated on the games of sizes comparable to the poker games used in the literature.

The remainder of the paper is structured as follows: In Section 1.1, we mention the most relevant literature. In Section 2, we formally describe perfect-information extensive-form games, Bayesian extensive games, and counterfactual regret minimization. In Section 3, we describe the BEG-specific version of CFR and analyze its complexity. We also hint at how some of the ideas can be extended to general EFGs; the details are presented in Appendix A. In Section 4, we illustrate our claims by presenting an empirical comparison of the general and BEG-specific versions of CFR on poker. In Section 5, we discuss the results and their implications.

1.1 Related Work

The topic of this text is tied to the informal concept of *incomplete information*. While some authors use this term interchangeably with *imperfect* information, some also make the following distinction: A player is said to have **incomplete information** if they are uncertain about the *rules* of the game (for example, legal actions, identity of other players, utility functions). In contrast, they are said to have imperfect information if they are uncertain about the current *state* of the game. In his seminal paper [8], Harsanyi explains the relationship between the two types of games and introduces (what is now typically called) Bayesian games as a formalization of strategic interaction under incomplete information. Sequential variants of this model are considered, for example, in [17, 6, 1] (under the names "Bayesian extensive game with observable actions", "multi-stage games with observed actions and incomplete information", and "games of incomplete information with observable actions" respectively).

2 Background

In this section, we give a formal definition of perfect-information extensive-form games and Bayesian extensive games and give several examples of BEGs, including poker. In Section 2.3, we describe the baseline CFR algorithm in the context of BEGs. To keep the formulas simple, we only present the results for two-player games. However, all of our results also hold for N-player games.[2]

2.1 Perfect Information EFGs

Perfect-information EFGs formalize the situation where several players make a sequence of publicly-observable decisions and possibly encounter a series of chance events, each of which affects the current state of the world and available actions. They assume some pre-specified set of terminal states over which the players have preferences.

Definition 1. *A **perfect information extensive-form game** is a tuple $\langle \mathcal{N}, \mathcal{A}, \mathcal{H}, \pi_{\mathrm{c}}, u \rangle$ where:*

- $\mathcal{N} := \{1, 2, \mathrm{c}\}$ *is the **player set** including **chance*** c.
- $\mathcal{A} = \prod_{i \in \mathcal{N}} \mathcal{A}_i$ *is non-empty finite set of **actions**.*
- $\mathcal{H}$ *is a finite tree on* $\mathcal{A}$ *(i.e., a set of finite sequences with extension relation* $\sqsubset$ *s.t.* $\forall g, h : h \in \mathcal{H}$ & $g \sqsubset h \implies g \in \mathcal{H}$*). Its elements are called **histories**.* $\mathcal{Z}$ *is the set of all leaves (terminal histories) of* $\mathcal{H}$.
- $\mathcal{A}(h) := \{a \in \mathcal{A} \mid ha \in \mathcal{H}\}$ *and* $\mathcal{A}_i(h) := \{a_i \mid a \in \mathcal{A}(h)\}$ *denotes **legal actions** (joint and player i's) at history h. We assume that* $\mathcal{A}(h) = \prod_{i \in \mathcal{N}} \mathcal{A}_i(h)$.
- *A player i is said to be **active** at h if* $|\mathcal{A}_i(h)| \geq 2$. *We assume that the game has no simultaneous moves, i.e., that there is most one active player at every h.*[3]
- $h \in \mathcal{H} \setminus \mathcal{Z} \to \pi_{\mathrm{c}}(h) \in \Delta\mathcal{A}_i(h)$ *is the **chance policy**.*
- $u : \mathcal{Z} \to \mathbb{R}^N$ *is the **utility function**.*

When indexing, we use the convention that i denotes one of the non-chance players, j denotes their opponent, and $-i$ denotes the pair (j, c).

2.2 Bayesian Extensive Games

BEGs can be viewed as an extension of perfect-information EFGs where the payoffs depend on the joint player-types which are initially drawn from a distribution known by all players[4]:

Definition 2 (Bayesian extensive game)**.** ***Bayesian extensive game** with publicly observable actions is a tuple* $G = \left\langle \Theta, \mu, \mathcal{N}, \mathcal{A}, \mathcal{H}, (\pi_{\mathrm{c}}^{\theta})_{\theta \in \Theta}, (u^{\theta})_{\theta \in \Theta} \right\rangle$*, where:*

- $\Theta = \Theta_1 \times \Theta_2$ *is the space of **player types**.*
- $\mu \in \Delta(\Theta)$ *is a common-knowledge **prior** over types.*

---

[2] Generalizing the results is straightforward — in most formulas, it suffices to replace j's reach probability by the product of reach probabilities of all opponents.

[3] We assume no simultaneous moves for an easier comparison with past work on CFR. This is, however, the *only* reason for this assumption — all arguments apply even with simultaneous moves (without requiring a change of notation).

[4] The model given by this definition is equivalent to taking the model from Section 8.2.3 of [6] (with possibly-correlated types, which the text also considers) and being explicit about modelling one player as "chance" with fixed strategy.

- *For every $\theta \in \Theta$, $\langle \mathcal{N}, \mathcal{A}, \mathcal{H}, \pi_c^\theta, u^\theta \rangle$ is an extensive-form game with perfect information.*

$\mathcal{S}_{\text{pub}}$ is called the **public tree** and its elements s_{pub} are called **public states** (or "public" histories). The set of its leaves is denoted $\mathcal{Z}_{\text{pub}}$. The tree of ("full") histories in G is $\mathcal{H} := \mathcal{S}_{\text{pub}} \times \Theta$. Each player in G only sees their own type θ_i and the current public state. Consequently, policies in G are defined as mappings from **information states** (infostates) $s_i = (s_{\text{pub}}, \theta_i) \in \mathcal{S}_{\text{pub}} \times \Theta_i$ to action probabilities $\pi_i(\,\cdot\,|s_i) \in \Delta\mathcal{A}_i(s_{\text{pub}})$.

We can illustrate BEGs using the example of Texas hold'em poker. Since the rules of play vary between different poker variants, we only describe the high-level commonalities. All poker variants start with each player drawing private cards (or card) from a deck. In the BEG terminology, this determines their type. The game progresses through several rounds of betting (that is, publicly-observable actions of players) between which the game reveals public cards (that is, publicly-observable chance actions). The game ends either when all players but one fold (give up), in which case all bets go to the remaining player, or in a showdown, where the player with the strongest combination of private and public cards receives all bets. (In other words, the utility is a function of the public state and θ). Note that since all cards are drawn from the same deck, representing poker as a BEG requires correlated types and type-dependent chance events.

Another example of a BEG is liar's dice (and its variants Bluff, Dudo, and others) [25]. While liar's dice looks different from poker at first glance, it is structurally very similar — it could be informally described as poker played with dice instead of cards. However, the game is even simpler than poker: Since each dice is rolled independently, the players' types are independent, and the chance policy doesn't depend on them. Finally, games like stratego [26] and battleship [24] can also be modelled as BEGs if we assume each player chooses their initial setup using a fixed (and known) probability distribution.

2.3 Counterfactual Regret Minimization

Counterfactual regret minimization is a popular self-play algorithm for imperfect-information games [27]. It approximates a Nash equilibrium by iteratively traversing the game tree and minimizing a particular notion of regret, called counterfactual regret, at every action at each decision point. (Where regret measures how much better off a player could have been if they changed all their actions at the given decision point $s_i = (s_{\text{pub}}, \theta_i)$ to a specific one, and everything else remained constant. The *counterfactual* part refers to assigning weights to iterations proportional to the probability of encountering s_i at the given iteration in the counterfactual scenario where i always selects actions that lead to s_{pub}.)

To describe CFR, we need the notion of a **counterfactual value** $V^\pi_{i,\text{cf}}(s_i)$ of an information state s_i:

$$V^\pi_{i,\text{cf}}(s_i) = V^\pi_{i,\text{cf}}(s_{\text{pub}}, \theta_i) := \sum\nolimits_{\theta_j \in \Theta_j} v^\pi_{i,\text{cf}}(s_{\text{pub}}, \theta)$$

$$v^\pi_{i,\text{cf}}(s_{\text{pub}}, \theta) := \sum_{s_{\text{pub}} \sqsubset z_{\text{pub}} \in \mathcal{Z}_{\text{pub}}} P_i^{\pi_i}(s_{\text{pub}} \to z_{\text{pub}}|\theta_i) P_j^{\pi_j}(z_{\text{pub}}|\theta_j) \cdot \cdot P_c(z_{\text{pub}}|\theta) u_i(z_{\text{pub}}|\theta),$$

where the P-symbols denote the probabilities that the given player takes all actions on the way to (or between) the given public state. for chance, this includes the probability of sampling θ:

$$P_j^{\pi_j}(z_{\text{pub}}|\theta_i) := \prod \left\{ \pi_j(a_j|s'_{\text{pub}}, \theta_j) \mid s'_{\text{pub}} a \sqsubset z_{\text{pub}} \right\}$$
$$P_c(z_{\text{pub}}|\theta) := \mu(\theta) \prod \left\{ \pi(a_c|s'_{\text{pub}}, \theta) \mid s'_{\text{pub}} a \sqsubset z_{\text{pub}} \right\}$$

$P_i^{\pi_i}(s_{\text{pub}} \to z_{\text{pub}}|\theta_i)$ is defined like $P_j^{\pi_j}(z_{\text{pub}}|\theta_i)$, except it also requires that $s'_{\text{pub}} \sqsupset s_{\text{pub}}$. We also define the corresponding **counterfactual infostate-action values**:

$$Q^\pi_{i,\text{cf}}(s_i, a_i) := V^{\pi|_{s_i \mapsto a_i}}_{i,\text{cf}}(s_i), \tag{2.1}$$

where $\pi|_{s_i \mapsto a_i}$ denotes the policy profile that coincides with π everywhere except for s_i, where it takes the action a_i with probability 1. Finally, the **counterfactual regret** at infostate s_i under policy π is

$$R^\pi_{i,\text{cf}}(s_i, a_i) := Q^\pi_{i,\text{cf}}(s_i, a_i) - V^\pi_{i,\text{cf}}(s_i).$$

For a more accessible and intuitive explanation of these concepts, we refer the reader to [21].

In practice, we compute counterfactual values of all infostates at once, in a single forwards- and backwards-pass of the game tree. The simplest version of the process, `Hist-CFVupdate`, is described in Algorithm 2: During the forward pass, incrementally compute the reach probabilities $P_j^{\pi_j}(z_{\text{pub}}|\theta_j)$ and $P_c(z_{\text{pub}}|\theta)$. During the backward-pass, incrementally compute $P_i^{\pi_i}(s_{\text{pub}} \to z_{\text{pub}}|\theta_i)$, such that upon reaching a history (s_{pub}, θ), we have all terms needed to calculate $v^\pi_{i,\text{cf}}(s_{\text{pub}}, \theta)$ and $v^{\pi|_{s_i \mapsto a_i}}_{i,\text{cf}}(s_{\text{pub}}, \theta)$. Finally, add these terms to $Q^i_{i,\text{cf}}(s_i, a_i)$ and $V^\pi_{i,\text{cf}}(s_i)$.

The last missing ingredient of CFR is translating counterfactual regrets into policy updates. The standard way of doing this is via the **regret matching** formula (`RM`):

$$\pi_i^{t+1}(a_i|s_i) := R^{t,+}_{i,\text{cf}}(s_i, a_i) \Big/ \sum\nolimits_{a'_i \in \mathcal{A}_i(s_{\text{pub}})} R^{t,+}_{i,\text{cf}}(s_i, a'_i)$$

where $R^{t,+}_{i,\text{cf}}(s_i, a'_i) := \max\{0, \sum_{k=1}^t R^{\pi^k}_{i,\text{cf}}(s_i, a'_i)\}$ and $\pi_i^{t+1}(a_i|s_i) := 1/|\mathcal{A}_i(s_{\text{pub}})|$ when the denominator is 0.

With all of these tools, defining **CFR** is straightforward (Algorithm 1): We initialize the algorithm with a uniformly random policy π^0. At each iteration, we calculate the counterfactual regrets of π^t for all infostates via `RegretUpdate` and use them to update the policy via regret matching. Finally, we return the average of the strategies π^t. For the purpose of this text, we refer CFR which uses the history-based implementation of `Hist-RegretUpdate` (Algorithm 2) as `Vanilla-CFR`.

Algorithm 1 `CFR` using a particular `RegretUpdate`

1: $\pi^0 \leftarrow$ uniform random policy
2: $R((s_{\text{pub}}, \theta_i), a_i) \leftarrow 0 \quad \forall s_{\text{pub}}, i, \theta_i, a_i \in \mathcal{A}_i(s_{\text{pub}})$
3: **for** $t = 0, \ldots, T-1$ **do**
4:     `RegretUpdate`(root)
5:     **for** all non-terminal s_{pub}, $i \neq$ c, and θ_i **do**
6:         $\pi_i^{t+1}(\,\cdot\,|s_{\text{pub}}, \theta_i) \leftarrow \texttt{RM}(R((s_{\text{pub}}, \theta_i), \,\cdot\,)$
    return $\bar{\pi} = \frac{1}{T}(\pi^1 + \cdots + \pi^T)$

Since `Vanilla-CFR` inspects every element of $\mathcal{H}$, its per-iteration run-time complexity is $O(|\mathcal{H}|)$. The memory complexity is lower-bounded by the number of infostates (because of the need to story the current policy). It can be higher if `Vanilla-CFR` stores the whole $\mathcal{H}$ in memory.

3 Public-State CFR

We saw that `Vanilla-CFR` works by iterating over the entire history tree of G. This section describes how to implement CFR by iterating over the public tree of G. This implementation performs most – or even all, in some domains – operations on the level of information states rather than EFG histories, leading to a significant reduction in time and memory complexity of the algorithm.

The key insight is that once we know the counterfactual values of leaf-infostates, the calculation for the remaining infostates gets nearly trivial. This is formally captured by the following (immediate) corollary of [21, Thm. 2]:

Proposition 3 (Backpropagation of counterfactual values)**.** *Let π be a policy profile and $\theta_i \in \Theta_i$ a type. Then we have the following for all terminal infostates $(z_{\text{pub}}, \theta_i)$ and non-terminal infostates $(s_{\text{pub}}, \theta_i)$:*

$$V^{\pi}_{i,\text{cf}}(z_{\text{pub}}, \theta_i) \;= \sum\nolimits_{\theta_j} P_j^{\pi_j}(z_{\text{pub}}|\theta_j) P_{\text{c}}(z_{\text{pub}}|\theta) u_i(z_{\text{pub}}|\theta)$$

$$Q^{\pi}_{i,\text{cf}}(s_{\text{pub}}, \theta_i, a_i) = \sum\nolimits_{a_{\text{-}i}} V^{\pi}_{i,\text{cf}}(s_{\text{pub}}\,(a_i, a_{\text{-}i}), \theta_i)$$

$$V^{\pi}_{i,\text{cf}}(s_{\text{pub}}, \theta_i) \;= \sum\nolimits_{a_i} \pi_i(a_i|s_{\text{pub}}, \theta_i)\; Q^{\pi}_{i,\text{cf}}(s_{\text{pub}}, \theta_i, a_i).$$

A corollary of Proposition 3 is that to calculate counterfactual value of a terminal infostate $(z_{\text{pub}}, \theta_i)$, we only need to know the chance-weighted utilities $\texttt{CWU}_i(z_{\text{pub}}) = (P_{\text{c}}(z_{\text{pub}}|\theta) u_i(z_{\text{pub}}|\theta))_{\theta \in \Theta}$ (these are a constant independent of the current policy) and the reach probabilities of those infostates of the opponent that are compatible with the current public state. This observation allows us to perform the CFR update by traversing the game tree on the level of public states, as described in the recursive procedure `PS-RegretUpdate` (Algorithm 3). Finally, we define **public-state CFR**, `PS-CFR`, as a variant of `CFR` that performs its updates using `PS-RegretUpdate`.

3.1 Complexity of CFR in BEGs

The main advantage of `PS-CFR` over `Vanilla-CFR` is its potential for increased practical and asymptotic efficiency.

On the practical side, the infostates compatible with a given public state are always indexed by the same set Θ_i and the list of legal is the same for all infostates compatible with the given public state. The upshot is that all parts of `PS-RegretUpdate` can be implemented as operations on vectors or matrices, making it suitable for parallelization. For example, the evaluation of a terminal public state z_{pub} (line 4) can be performed for all infostates $(z_{\text{pub}}, \theta_i)$ at once by multiplying the matrix $\texttt{CWU}_i(z_{\text{pub}})$ with the vector $P_j(z_{\text{pub}}|\cdot)$.

On the asymptotic side, the limiting factor is the need to store and update the policy (at least for tabular implementations) — this means that the time and space complexity of one `PS-CFR` operation cannot be lower than $|\mathcal{I}| := \bigcup_{i=1}^{2} |\mathcal{S}_{\text{pub}} \times \Theta_i|$, the number of information states in the game. However, this still leaves a lot of space for improvement since the complexity of `Vanilla-CFR` (and a naive implementation of `PS-CFR`) is $O(|\mathcal{H}|)$, which can be as large as $O(|\mathcal{I}|^2)$. The primary method for achieving an asymptotic speedup is by implementing a domain-specific evaluation of terminal public states [9] (for the proof, see Appendix B):

Theorem 4 (Complexity of `PS-CFR`)**.** *For a BEG G:*

1. *The time and space complexity of one iteration of `PS-CFR` is $O(|\mathcal{H}|)$.*
2. *There are domains, incl. poker, where the time and space complexity of one iteration of `PS-CFR` is $O(|\mathcal{I}|)$.*

Poker serves as a good illustration of (2): The key insight is that if we want to know the expected utility in poker, it is unnecessary to know the probability of each card combination (hand) that the opponent could have. Instead, we only need to know the probability that their hand is weaker than ours. Moreover, if our hand changed to a stronger one, the probability of the opponent's hand being weaker would only change by the probability of the opponent holding cards weaker than our new hand but stronger than our original hand. This observation allows us to re-use most of the computation, bringing the per-public-state complexity from $|\Theta_1||\Theta_2|$ to $|\Theta_1| + |\Theta_2|$.

3.2 PS-CFR beyond BEGs

We now present the high-level ideas which allow us to extend `PS-CFR` to general imperfect-information games and compare this more general case with the specialized BEG version. The formal definitions and details of the algorithm are explained in Appendix A.

We can informally introduce **imperfect-information extensive-form games** (EFGs) by contrasting them to BEGs. While the underlying structure for both models is a perfect-information game, they differ in the type of partial observability they introduce: In a BEG, the players are uncertain about the *payoffs* at the end of the game. In contrast, the players in an EFG are instead uncertain about the game's current *state*. EFGs formalize this idea by partitioning the histories into **information sets** (infosets) for each player and requiring that the player's policy is a function of the current infoset. It follows that BEGs can be viewed as a special case of EFGs — indeed, every BEG can be modelled as an EFG by starting the game with a chance node that determines

Algorithm 2 `Hist-RegretUpdate`(h)

let s, θ be s.t. $h = (s, \theta)$ ▷ Retrieve information in h
if $h = \text{root}$ **then**
  $P_i(s, \theta_i) \leftarrow 1 \quad \forall i, \theta_i$
if $z = h \in \mathcal{Z}$ **then**
  return $(\, P_j(s|\theta_j) P_c(s|\theta) u_i(z) \,)_{i=1,2}$
else
  $v_i \leftarrow 0 \quad \forall i$
  for every child $g = (sa, \theta)$ of $h = (s, \theta)$ **do**
    $P_i(sa, \theta_i) \leftarrow P_i(s, \theta_i) \pi_i(a|s, \theta_i) \quad \forall i$
    $q(a) \leftarrow$ `Hist-RegretUpdate`(g)
    $v_i \mathrel{+}= \pi_i(a|s, \theta_i)\; q(a)_i \quad \forall i$
  $R((s, \theta_i), a) \mathrel{+}= q(a)_i - v_i \quad \forall i, a \in \mathcal{A}_i(h)$
  return v

Algorithm 3 `PS-RegretUpdate`(s)

Require: $\texttt{CWU}_i(z)$ for all z and i
if $s = \text{root}$ **then**
  $P_i(s, \theta_i) \leftarrow 1 \quad \forall i, \theta_i$
if $z = s \in \mathcal{Z}_{\text{pub}}$ **then**
  $\texttt{CFV}(z, \theta_i) \leftarrow \sum_{\theta_j \in \Theta_j} P_j(z|\theta_j) \texttt{CWU}_i(z)(\theta) \quad \forall i, \theta_i$
else
  for every child sa of s **do**
    $P_i(sa, \theta_i) \leftarrow P_i(s, \theta_i) \pi_i(a_i|s, \theta_i) \quad \forall i, \theta_i$
    `PS-RegretUpdate`(s)
  for $\forall i, \theta_i$, and $a_i \in \mathcal{A}_i(s)$ **do**
    $\texttt{CFQ}(s, \theta_i, a_i) \leftarrow \sum_{a_{\text{-}i} \in \mathcal{A}_{\text{-}i}(s)} \texttt{CFV}(sa, \theta_i)$
    $\texttt{CFV}(s, \theta_i) \leftarrow \sum_{\mathcal{A}_i(s)} \pi_i(a_i|s, \theta_i)\; \texttt{CFQ}(s, \theta_i, a_i)$
    $R((s, \theta_i), a_i) \leftarrow \texttt{CFQ}(s, \theta_i, a_i) - \texttt{CFV}(s, \theta_i)$

Figure 1: Implementations of the `Vanilla-CFR` regret update on the history tree and the `PS-CFR` regret update on the tree of public states. For brevity, we drop the "pub" index for public states s_{pub}.

each player's type and grouping together the histories which only differ in the other players' type. The EFG analogue of a public state is a **public *set*** [9, 11]; the collection of all public sets forms an additional partition that is refined by each player's information partition, such that if a history belongs to some public set, this fact can be considered common knowledge. An infoset I is compatible with a public set S if $I \subset S$. I is compatible with an infoset J if $I \cap J \neq \emptyset$.

This analogy between BEGs and EFGs makes the high-level description of `PS-CFR` for EFGs straightforward: The algorithm is analogous to its BEG-version, except that where the original version iterates over public states, resp. types, the EFG-version iterates over public sets, resp. infosets compatible with the current public set.

However, two subtle-yet-important complications make the EFG-version of `PS-CFR` more conceptually difficult and less amenable to an efficient implementation. First, running the algorithm requires being able to (a) view the public partition as a tree (that is, to find parents and children of public sets) and (b) generate a list of infosets compatible with each public set. While both operations can be performed on-demand and efficiently in BEGs, it is currently unclear how to run them easily in *general* EFGs. (While we do not have decisive arguments on this subject, we expect there will be games – such as blind chess – where these operations are *inherently* costly.) Second, BEGs have a very homogeneous structure, which allows us to write many parts of the `PS-CFR` algorithm as vector or matrix operations. As we will see, this is not true for general EFGs, which can have a very irregular structure and variables of disparate dimensions.

To understand the complications in general EFGs, it is helpful to realize that BEGs are a *very* special case of EFGs, enjoying several simplifying properties not shared by EFGs. First, the structure of information is homogeneous across the whole public tree in a BEG — the public state is always common knowledge, each player's type is private, and no information is ever hidden from everybody. In contrast, in general EFGs, (1) players may have no information at all, perfect information, or anything between, (2) their information can be entirely overlapping, entirely disjoint, or anything between, and (3) the distribution of information can change drastically throughout the game. In particular, the number of infosets compatible with a given public set can vary significantly between different public sets, and different infosets of one player might be compatible with different infosets of another player. Second, actions in a BEG are always fully observable, which is often not the case in general EFGs. Finally, even the set of legal actions might vary depending on the private information in EFGs, while it only depends on public information in BEGs.

4 Empirical Evaluation

In the previous section, we have shown that in poker, `PS-CFR` with a domain-specific evaluation of terminal states has asymptotically lower run-time and memory complexity than `Vanilla-CFR`. We also argued that `PS-CFR` is likely to be more efficient than `Vanilla-CFR` even without the domain-specific optimizations. We now compare the performance of these algorithms empirically.

We evaluate the results on a subgame of no-limit Texas hold'em poker. Specifically, we use the subgame after the last public card is dealt (that is, a river subgame), with public cards (9s, 7c, 5s, 4h, 3c), pot size 200, and uniform distribution over private cards. To make the computation tractable, we use the (fold, call, pot, all-in) action-abstraction, which results in a subgame that has 61,000,831 states (i.e., histories) and 21,620 decision-points (i.e., active-player infosets).

All algorithms are implemented using the open-source library OpenSpiel [13] (and we plan to incorporate them into the master branch). We implemented two versions

Algorithm	Setup time	1000 iterations	One iteration	Memory used
`Vanilla-CFR`	2.92 min	5.48 h	19.73 s	22 GB
`PS-CFR`	2.36 s	2.89 min	173.15 ms	736 MB
`PS-CFR (domain-sp)`	2.82 s	1.42 min	85.05 ms	526 MB
`Vanilla-CFR (memory-eff)`	1.57 min	25.75 h	92.71 s	*292 MB*[5]

Table 1: Comparison of `Vanilla-CFR` and `PS-CFR` on a river subgame of no-limit Texas hold'em poker.

of public-state CFR — the baseline version `PS-CFR` applicable to any BEG and a poker-specific version `PS-CFR (domain-sp)` whose terminal-state evaluation run-time is linear (rather than quadratic) in the number of infosets [9]. For vanilla CFR, we used a version already present in OpenSpiel. We refer to this version as `Vanilla-CFR (memory-eff)` since it only maintains a small portion of the game tree at any given time, resulting in slower child-retrieval but low memory usage. To make the algorithm more directly comparable to our implementation of public-state CFR, we also implemented a version that keeps the structure needed for child-retrieval in memory. We refer to this version simply as `Vanilla-CFR`.

We ran 1000 iterations of each algorithm and measured the resulting memory usage, the time required for initialization, and the subsequent time needed to run the 1000 iterations (Table 1). First, we see that the poker-specific version of `PS-CFR` takes slightly longer to initialize, but afterwards only requires 70% of the memory and runs twice as fast. Among the two versions of vanilla CFR, `Vanilla-CFR (memory-eff)` is roughly five times slower than `Vanilla-CFR` but uses ~75x less memory. Most importantly, we see that both versions of vanilla CFR are extremely slow compared to `PS-CFR`: The faster version requires ~200x more time and ~40x more memory than `PS-CFR (domain-sp)`, while the slower version requires ~1000x more time and a similar amount of memory.

5 Conclusion

We have recently seen a lot of progress around counterfactual regret minimization. However, while many works aim to solve general (two-player zero-sum) imperfect information games — typically formalized as extensive-form games — their empirical evaluation tends to only consider poker, liar's dice, or other games whose structure is near-identical to poker. As a result, existing implementations of most CFR-based algorithms use many optimizations which rely on non-generalizable poker-specific assumptions. These optimizations are often so extensive that it might be more appropriate to view the implemented algorithm as distinct from the CFR described in [27].

We would like to bring attention to a different formalization called Bayesian extensive games (BEGs) [17, 6, 1], which is natural for describing poker and other similar games. We have shown that many elements of the poker-specific implementation of CFR generalize to this class of games. While the classical version of CFR (`Vanilla-CFR`) traverses the game tree on the level of histories, the BEG version traverses it on the level of public states; we thus called it public-state CFR, or `PS-CFR` for short.

We see at least three benefits of the public-state formulation of CFR. The first is conceptual: Many recent extensions of CFR (e.g., [2, 19]) heavily rely on decomposition and public states. Since `PS-CFR` is also formulated in terms of public states, `PS-CFR` serves as a much more suitable basis for these extensions than `Vanilla-CFR`. Second, even though `PS-CFR` has – in general – the same asymptotic complexity as `Vanilla-CFR`, it naturally lends itself to an efficient implementation using vectors and matrices, which leads to a significant practical speedup over `Vanilla-CFR`, especially on GPUs. Indeed, we saw that `PS-CFR` only requires around three minutes and 700 MB to solve a river subgame of no-limit Texas hold 'em poker while `Vanilla-CFR` needed around 5.5 hours and 20 GB. Third, we saw that many domains allow for a more efficient implementation of `PS-CFR` (more specifically, of the evaluation of terminal states), which can lead to an improvement in CFR's *asymptotic* complexity. In the case of poker, this reduces the complexity of a single iteration of CFR from $O(|\mathcal{H}|)$ (the number of game-states) to $O(|\mathcal{I}|)$ (the number of decision points).

We have also shown that the ideas behind `PS-CFR` can be extended to general EFGs. Like the BEG version of `PS-CFR`, the EFG version is still suitable for integration of decomposition-based methods and amenable to domain-specific improvements. However, it is conceptually more complicated, does not, in general, lend itself to vectorized implementation, and requires a possibly-costly initialization (to build the public tree and connect it to the information-state trees). We have implemented both versions of the algorithm and the necessary initialization methods, and both will be made available in the open-source library OpenSpiel [13].

We believe these results have several implications. First, to the extent that we are interested in sequential Bayesian games, it would be beneficial to explicitly adopt the BEG model. This will make the results more understandable and replicable. Moreover, it might make developing and implementing new ideas easier. Second, to advance state of the art in general imperfect-information games, we should work with more domains than just

[5] The memory requirements of `Vanilla-CFR (memory-eff)` should be compared to an analogous low-memory version of `PS-CFR`, which we did not implement. However, the most memory-expensive part of `Vanilla-CFR (memory-eff)` is storing the current strategy. We thus predict that the hypothetical `PS-CFR (memory-eff)` would require a near-identical amount of memory while still being two orders of magnitude faster.

poker and BEGs. We believe clarifying the challenges present in more general domains and establishing appropriate benchmarks to be important next steps in this direction. Finally, in domains that cannot easily be cast as BEGs — or with BEGs that are not as amenable to action-abstraction as poker — we should recalibrate our expectation on performance and not require new algorithms to immediately scale to poker-sized domains.

Acknowledgments

This work was supported by Czech science foundation grant no. 18-27483Y. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] P. Battigalli and M. Siniscalchi. Rationalization and incomplete information. *Advances in Theoretical Economics*, 3(1), 2003.

[2] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong. Combining deep reinforcement learning and search for imperfect-information games. *arXiv preprint arXiv:2007.13544*, 2020.

[3] N. Brown, A. Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. *arXiv preprint arXiv:1811.00164*, 2018.

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

[5] N. Burch, M. Johanson, and M. Bowling. Solving imperfect information games using decomposition. In *AAAI*, pages 602–608, 2014.

[6] D. Fudenberg and J. Tirole. Game theory, 1991. *Cambridge, Massachusetts*, 393(12):80, 1991.

[7] R. Gibson, M. Lanctot, N. Burch, D. Szafron, and M. Bowling. Generalized sampling and variance in counterfactual regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[8] J. C. Harsanyi. Games with incomplete information played by “Bayesian” players, I–III, part I. the basic model. *Management science*, 14(3):159–182, 1967.

[9] M. Johanson, K. Waugh, M. Bowling, and M. Zinkevich. Accelerating best response calculation in large extensive games. In *IJCAI*, volume 11, pages 258–265, 2011.

[10] M. Johanson, M. Zinkevich, and M. Bowling. Computing robust counter-strategies. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 721–728, 2007.

[11] V. Kovařík, M. Schmid, N. Burch, M. Bowling, and V. Lisỳ. Rethinking formal models of partially observable multiagent decision making. *Artificial Intelligence*, page 103645, 2021.

[12] M. Lanctot, R. Gibson, N. Burch, M. Zinkevich, and M. Bowling. No-regret learning in extensive-form games with imperfect recall. *arXiv preprint arXiv:1205.0622*, 2012.

[13] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[14] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pages 1078–1086, 2009.

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

[16] O. Morgenstern and J. Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1953.

[17] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1994.

[18] M. Ponsen, M. Lanctot, and S. De Jong. MCRNR: Fast computing of restricted Nash responses by means of sampling. In *Workshops at the twenty-fourth AAAI conference on artificial intelligence*, 2010.

[19] M. Schmid, N. Burch, M. Lanctot, M. Moravcik, R. Kadlec, and M. Bowling. Variance reduction in Monte Carlo counterfactual regret minimization (VR-MCCFR) for extensive form games using baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2157–2164, 2019.

[20] M. Schmid, M. Moravcik, N. Burch, R. Kadlec, J. Davidson, K. Waugh, N. Bard, F. Timbers, M. Lanctot, Z. Holland, et al. Player of games. *arXiv preprint arXiv:2112.03178*, 2021.

[21] D. Seitz, V. Kovařík, V. Lisý, J. Rudolf, S. Sun, and K. Ha. Value functions for depth-limited solving in imperfect-information games beyond poker. *arXiv preprint arXiv:1906.06412*, 2019.

[22] M. Šustr, V. Kovařík, and V. Lisý. Monte Carlo continual resolving for online strategy computation in imperfect information games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 224–232. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[23] O. Tammelin, N. Burch, M. Johanson, and M. Bowling. Solving heads-up limit Texas hold’em. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[24] Wikipedia. Battleship (game). https://en.wikipedia.org/wiki/Battleship_(game), 2021. accessed: 2021-11-11.

[25] Wikipedia. Dudo. https://en.wikipedia.org/wiki/Dudo, 2021. accessed: 2021-11-11.

[26] Wikipedia. Stratego. https://en.wikipedia.org/wiki/Stratego, 2021. accessed: 2021-11-11.

[27] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.

A PS-CFR in General EFGs

In this section, we describe an implementation of the public-state CFR algorithm for extensive-form games, i.e., for a class that is much more general than Bayesian extensive games. On the high level, the algorithm is very similar to `PS-CFR` for BEGs. However, we will see that the low-level details are more complicated and less amenable to efficient implementation.

To model general imperfect-information games, we endow perfect-information games with an additional structure – information sets and public sets – which captures the histories which are indistinguishable from the point of view of some player, resp. an external observer who doesn't have access to any private information. (The role of the latter is similar to that of public states in BEGs.)

Definition 5. *An* ***imperfect-information extensive-form game*** *(EFG) is a pair $\langle \Gamma, \mathcal{I} \rangle$, where Γ is an N-player perfect-information extensive-form game and $\mathcal{I} = (\mathcal{I}_1, \ldots, \mathcal{I}_N, \mathcal{I}_{\text{pub}})$ is a collection of partitions of $\mathcal{H}$, where each $\mathcal{I}_i$*

- *is a refinement[6] of $\mathcal{I}_{\text{pub}}$ and*
- *provides enough information to identify i's legal actions.[7]*

The elements of $\mathcal{I}_i$ and $\mathcal{I}_{\text{pub}}$ are called ***information sets (infosets)*** *and* ***public sets****.*

To avoid various pathologies, we additionally require perfect recall and "no thick infosets or public sets": An EFG is with **perfect recall** if for each $g, h \in I_i \in \mathcal{I}_i$, i's action-infoset histories[8] corresponding to g and h coincide. An EFG is said to not have **thick public sets** if no element of $\mathcal{I}_{\text{pub}}$ (and hence of $\mathcal{I}_i$) contains both some h and its strict extension.

The definition of counterfactual values in an EFG is similar to the BEG definition: A policy π_i in an EFG maps i's infosets to probability distributions over actions legal at that infoset. A **counterfactual value of an infoset** $I_i \in \mathcal{I}_i$ under policy profile π is defined as

$$V^{\pi}_{i,\text{cf}}(I_i) := \sum_{h \in I_i} v^{\pi}_{i,\text{cf}}(h), \text{ where}$$

$$v^{\pi}_{i,\text{cf}}(h) := \sum_{h \sqsubset z \in \mathcal{Z}} (P^{\pi_i}_i(h \to z)) \prod_{j \neq i,\text{c}} P^{\pi_j}_i(z) P_\text{c}(z) u_i(z)$$

and where the P-symbols denote the product of action probabilities $\pi_i(a|I_i)$ (resp. $\pi_\text{c}(a|h)$) for all actions taken on the way to the given terminal state z (resp. along the trajectory from h to z). Since i's only depend on the infoset to which a history belongs, we denote $P^{\pi_i}_i(I_i) := P^{\pi_i}_i(h)$ (where h is an arbitrary element of I_i), and similarly for $P^{\pi_i}_i(I_i \to J_i)$. As in BEGs, we define the counterfactual q-values as $Q^i_{i,\text{cf}}(I_i, a_i) := V^{\pi|_{I_i \mapsto a_i}}_{i,\text{cf}}(I_i)$, where $\pi|_{I_i \mapsto a_i}$ is like π, except i takes a_i at I_i.

The computation of counterfactual values in EFGs is similar to Proposition 3 (and also follows from [21, Thm. 2]):

Proposition 6 (Backpropagation of counterfactual values in EFGs)**.** *Let π be a policy profile in an EFG. When I_i is terminal infoset s.t. $I_i \subset Z \in \mathcal{I}_{\text{pub}}$, we have*

$$V^{\pi}_{i,\text{cf}}(I_i) = \sum_{(I_j)_j \in \prod_{j \neq i,\text{c}} \mathcal{I}_j(Z)} \prod_{j \neq i,\text{c}} P^{\pi_j}_j(I_j) \left(\sum_{z \in I_1 \cap \dots \cap I_N} P_\text{c}(z) u_i(z) \right)$$

When I_i is non-terminal and $a_i \in \mathcal{A}_i(I_i)$, we have

$$Q^{\pi}_{i,\text{cf}}(I_i, a_i) = \sum \left\{ V^{\pi}_{i,\text{cf}}(J_i) \;\middle|\; J_i \in \mathcal{I}_i \text{ s.t. } (\forall h \in J_i) \right.$$
$$\left. (\exists g \in I_i)(\exists a_{\text{-}i}) : h = g(a_i, a_{\text{-}i}) \right\}$$
$$V^{\pi}_{i,\text{cf}}(I_i) = \sum_{a'_i \in \mathcal{A}_i(I_i)} \pi_i(a'_i|I_i) \; Q^{\pi}_{i,\text{cf}}(I_i, a'_i).$$

As in the BEG, we use Proposition 6 to define `PS-CFR` on EFGs. However, we first need to make sure that the information partitions and the public partition support the following operations:

- For each infoset $I \in \mathcal{I}_i$, we can get the previous infoset J encountered by i and the action a'_i taken by i at J.
- For each public set S, we can get[9]
  - the previous[10] public set S',
  - the list of all child public-sets of S,
  - the list $\mathcal{I}_i(S) := \{I_i \in \mathcal{I}_i \mid I_i \subset S\}$ of i's infosets compatible with S.

Note that while the public-state operations were mostly trivial in BEGs, this is no longer true in general EFGs and we expect that in some domains, fast implementation of these operations will be impossible.

The `PS-CFR` algorithm for EFGs is defined as follows:

[6] Recall that $\mathcal{P}$ is a partition of a set X if $\bigcup \mathcal{P} = X$ and $(\forall P, P' \in \mathcal{P}) : P \neq P' \implies P \cap P' = \emptyset$. $\mathcal{P}$ is a refinement of a partition $\mathcal{Q}$ if each $P \in \mathcal{P}$ is a subset of exactly one $Q \in \mathcal{Q}$.

[7] That is, for every $I_i \in \mathcal{I}_i$, $\mathcal{A}(h)$ is the same for all $h \in I_i$.

[8] Where i's action-infoset history corresponding to h is the sequence of infosets encountered by i and actions taken by i on the way to h.

[9] The reason for using S instead of P to denote public sets is to avoid notation clash with reach probabilities.

[10] That is, the public set S' for which every $h \in S$ is an immediate successor of some $g \in S'$.

1. Initialization:
   (a) Build the data structures necessary for the above operations.
   (b) For each terminal $Z \in \mathcal{I}_{\text{pub}}$, precompute the chance-weighted payoff matrix $\texttt{CWU}_i(Z) =$
$$\left(\sum_{h \in I_1 \cap \cdots \cap I_N} P_{\text{c}}(h) u_i(h) \right)_{(I_1,\ldots,I_N) \in \mathcal{I}_1(Z) \times \cdots \times \mathcal{I}_N(Z)} .$$
   (c) Define π^0 as the uniformly-random policy.
2. For each $t = 0, \ldots, T-1$:
   (a) Call EFG-PS-CFVupdate(root) (defined below) to update cumulative regrets $R(\cdot|S, I_i)$.
   (b) Update strategy: For every non-terminal S, $i \neq$ c, and $I_i \in \mathcal{I}_i(S)$, obtain $\pi_i^{t+1}(\cdot|s_{\text{pub}}, \theta_i)$ by using RM on $R(\cdot|S, I_i)$.
3. Return $\bar{\pi} = \frac{1}{T}(\pi^1 + \cdots + \pi^T)$.

The recursive procedure EFG-PS-CFVupdate takes a public set S as input, and – denoting $\pi := \pi^t$ – is defined as follows:

(1) Calculate reach probabilities:
- If S is the root, set $P_i(S, I_i) = 1$ for all $I_i \in \mathcal{I}_i(S)$.
- Otherwise, set[11] $P_i(S, I_i) = P_i(S', J_i)\pi_i(a_i'|J_i)$ for all $I_i \in \mathcal{I}_i(S)$.

(2) Calculate counterfactual values:
- If $S = Z$ is terminal, set the following for every i and $I_i \in \mathcal{I}_i(S)$, compute $\texttt{CFV}(Z, I_i) =$
$$\sum_{(I_j)_j \in \prod_{j \neq i,\text{c}} \mathcal{I}_j(Z)} \left(\prod_{j \neq i,\text{c}} P_j(S, I_j) \right) \texttt{CWU}_i(Z)(I_1, \ldots, I_N).$$
- Otherwise, set $\texttt{CFQ}_i(S, I_i, a_i) := 0$ for all i, $I_i \in \mathcal{I}_i(S)$, and $a_i \in \mathcal{A}_i(I_i)$, run EFG-PS-CFVupdate$(S')$ for every child S' of S, and set
$$\texttt{CFV}_i(S, I_i) := \sum_{a_i \in \mathcal{A}_i(I_i)} \pi_i(a_i|I_i) \texttt{CFQ}_i(S, I_i, a_i).$$

(3) Back-propagate counterfactual values:
- Add $\texttt{CFV}_i(S, I_i)$ to $\texttt{CFQ}_i(S', J_i, a_i')$ for every i and $I_i \in \mathcal{I}_i(S)$.
- Use them to calculate the corresponding regrets.

B Proofs

Theorem 4 (Complexity of PS-CFR)**.** *For a BEG G:*
1. *The time and space complexity of one iteration of PS-CFR is $O(|\mathcal{H}|)$.*
2. *There are domains, incl. poker, where the time and space complexity of one iteration of PS-CFR is $O(|\mathcal{I}|)$.*

Proof. (1) In non-terminal public states, the time and space complexity of RegretUpdate (without the recursion) is $O(|\mathcal{I}|)$. In terminal public states, the algorithm needs to – in general – store and inspect all $|\Theta|$ elements of the matrix $\texttt{CWU}_i(z_{\text{pub}})$. The general time and space complexity is thus $O(|\mathcal{S}_{\text{pub}} \times \mathcal{I}| + |\mathcal{S}_{\text{pub}} \times \Theta|) = O(|\mathcal{S}_{\text{pub}} \times \Theta|) = O(|\mathcal{H}|)$.

(2) If we reduce the complexity of evaluating terminal public states from $|\Theta_1|\,|\Theta_2|$ to some $E \geq \sum_{i=1}^2 |\Theta_i|$, we could bring the overall complexity down to $O(|\mathcal{S}_{\text{pub}} \times \mathcal{I}| + |\mathcal{S}_{\text{pub}}| \cdot E) = O(|\mathcal{S}_{\text{pub}}| \cdot E)$. By [9], such more-efficient evaluation of terminal public states is possible in a number of domains, and poker in particular admits $E = \sum_{i=1}^2 |\Theta_i|$. In poker, we thus have $|\mathcal{S}_{\text{pub}}| \cdot E = |\mathcal{S}_{\text{pub}}| \cdot \sum_{i=1}^2 |\Theta_i| = |\mathcal{I}|$, which concludes the proof. (In poker, this requires a one-time investment to sort the private cards in each state based on their strength [9]. However, this cost can be amortized across all of the iterations.) □

[11] Where S', J_i, and a_i' are the parent of S, parent of I_i, and the last action taken by i.