

# Safe Subgame Resolving for Extensive Form Correlated Equilibrium

Chun Kai Ling,<sup>1</sup> Fei Fang,<sup>1</sup>

<sup>1</sup> Carnegie Mellon University  
chunkail@cs.cmu.edu, feif@cs.cmu.edu

## Abstract

Correlated Equilibrium is a solution concept that is more general than Nash Equilibrium (NE) and can lead to outcomes with better social welfare. However, its natural extension to the sequential setting, the *Extensive Form Correlated Equilibrium* (EFCE), requires a quadratic amount of space to solve, even in restricted settings without randomness in nature. To alleviate these concerns, we apply *subgame resolving*, a technique extremely successful in finding NE in zero-sum games to solving general-sum EFCEs. Subgame resolving refines a correlation plan in an *online* manner: instead of solving for the full game upfront, it only solves for strategies in subgames that are reached in actual play, resulting in significant computational gains. In this paper, we (i) lay out the foundations to quantify the quality of a refined strategy, in terms of the *social welfare* and *exploitability* of correlation plans, (ii) show that EFCEs possess a sufficient amount of independence between subgames to perform resolving efficiently, and (iii) provide two algorithms for resolving, one using linear programming and the other based on regret minimization. Both methods guarantee *safety*, i.e., they will never be counterproductive. Our methods are the first time an online method has been applied to the correlated, general-sum setting.

## Introduction

Correlation between players is a powerful tool in game theory. The *Correlated Equilibrium* (CE) is an equilibrium that allows for players to coordinate actions with the aid of a mediator or a randomized correlation device, and is known to allow for outcomes which lead to a significantly higher social welfare as compared to solution concepts which require independent play, such as Nash Equilibrium (NE), on top of being computationally more tractable. In a CE, the mediator recommend actions privately to the players according to a probability distribution over joint actions that is known to all players, and the players have no incentive to deviate from the recommended action if they are perfectly rational. A natural extension of CE to *extensive form games* (EFG) is the *Extensive Form Correlated Equilibrium* (EFCE), where players are recommended actions at each decision point (Von Stengel and Forges 2008). Unfortunately, solving and storing an EFCE typically requires space that is quadratic in the size

of the game tree. This is a significant barrier towards solving large games: for example, storing an EFCE for a game of Battleship (Farina et al. 2019a) with a grid size of  $3 \times 2$  requires a vector with more than  $10^8$  entries.

Over the last decade, a technique known as subgame resolving has gathered much attention amongst those looking to solve large games. The idea behind subgame resolving is to adopt a simple blueprint strategy at the beginning, and to compute refinements of the strategy in an *online* manner only when the game has entered a *subgame*. This means that one need not compute strategies in branches of the game which were never reached in actual play, just like with limited-depth search in perfect information games like chess. Rising into prominence because of the successes of superhuman-level poker bots such as *Libratus* (Brown and Sandholm 2017, 2018), subgame resolving has since been studied from other angles (Zhang and Sandholm 2021), extended to other equilibrium concepts (Ling and Brown 2021) and applied in practice to, multiplayer games like Hanabi (Lerer et al. 2020) and Diplomacy (Gray et al. 2021). However, subgame resolving has primarily been applied to the zero-sum or cooperative settings, with few inroads in the correlated setting, where the objective is to get players to coordinate despite potentially having misaligned interests.

In this paper, we introduce subgame resolving for EFCE. Instead of announcing the full correlation plan that specifies the probability of recommending different actions at each decision point, the mediator computes the EFCE strategies online. Conceptually, it can be viewed as having the mediator publish the algorithm of choosing recommended actions, and the algorithm is designed in a way such that the rational players will have no incentive to deviate from the recommended actions. Our contributions are twofold. First, we lay out the framework for safe subgame resolving for EFCE in terms of the exploitability of a correlation plan with respect to a correlation blueprint. Second, we show that for games without chance, the structure of the polytope of correlation plans contains a sufficient level of independence between subgames to facilitate independent solving. Third, we provide two refinement algorithms, the first based on a modification of the linear program (LP) of Von Stengel and Forges (2008), and the second utilizing a recent and more efficient method based on regret minimization (Farina et al. 2019b). To the best of our knowledge, this is the first instance of sub-

game resolving being applied to the correlated setting. We experimentally show its scalability in benchmark games.

## Background and Related Work

Let  $\mathcal{G}$  be a 2-player extensive-form-game *without chance*. This is represented by a finite game tree: nodes represent game states, belonging to either player  $P_1$  or  $P_2$ , while actions are represented by edges directed down the tree. To represent imperfect information,  $\mathcal{G}$  is supplemented with *information sets* (infosets)  $I_i \in \mathcal{I}_i, i \in [2]$ , which are collection of states belonging to but are indistinguishable to  $P_i$ . States in the same infoset contain the same actions  $a_i \in \mathcal{A}(I_i)$ . We denote by  $ha$  the state that is reached immediately after taking action  $a$  at state  $h$ . We say that state  $h$  precedes ( $\sqsupseteq$ )  $h'$  if  $h \neq h'$  and  $h'$  is a descendent of  $h$  in the game tree, and use the notation  $h \sqsubseteq h'$  when allowing  $h = h'$ . We assume players have perfect recall, that is, players never forget past observations and past actions. The set of terminal states  $\mathcal{L}$  are known as *leaves*. Each leaf is associated with utilities received by each player  $u_i(h)$ . For a given leaf  $h$ , the *social welfare* is given by  $u_1(h) + u_2(h)$ .

We define the set of *sequences* for  $P_i$  as the set  $\Sigma_i := \{(I, a) : I \in \mathcal{I}_i, a \in \mathcal{A}(I)\} \cup \{\emptyset\}$ , where  $\emptyset$  is known as the *empty sequence*. For any infoset  $I_i \in \mathcal{I}_i$ , we denote by  $\sigma(I)$  the *parent sequence* of  $I$ , which is defined as the (unique) sequence which precedes  $I$  from the root to any node in  $I$ ; if no such sequence exists, then  $\sigma(I) = \emptyset$ . Sequences in  $\Sigma_i$  form a partial order; for sequences  $\tau = (I, a), \tau' = (I', a') \in \Sigma_i$ , we write  $\tau \prec \tau'$  if there exists states  $ha, h' \in I'$  belonging to  $P_i$  such that  $ha \sqsubseteq h'$ , and write  $\tau \preceq \tau'$  if allowing  $\tau = \tau'$ . If in addition,  $\sigma(I') = \tau$ , we say that  $\tau'$  is an immediate successor of  $\tau$  and write  $\tau \prec_1 \tau'$ . Since the game has no chance, each leaf  $h \in \mathcal{L}$  is uniquely identified by a pair of sequences  $(\sigma_1, \sigma_2)$ . With a slight abuse of notation we write  $(\sigma_1, \sigma_2) \in \mathcal{L}$ , and denote corresponding player payoffs and social welfare by  $u_i(\sigma_1, \sigma_2)$  and  $u(\sigma_1, \sigma_2)$ .

**Sequence-form strategies** In the sequence form, a (mixed) strategy for  $P_i$  is compactly represented by a vector  $x_i$ , indexed by the sequences  $\sigma = (I, a) \in \Sigma_i$ . The entry  $x_i[\sigma]$  contains the *product* of the probabilities of  $P_i$  taking actions from the root to information set  $I^1$ , including  $a$  itself, with the base case given by  $x_i[\emptyset] = 1$ . Hence, valid sequence-form strategies must satisfy the ‘flow’ constraints; for every  $I \in \mathcal{I}_i$ , we have  $\sum_{a \in \mathcal{A}(I)} x_i[(I, a)] = \sigma(I)$ . Sequence-form strategies have size roughly equal to the number of actions of the player, while flow constraints can be seen as a generalization of the sum-to-one constraints for strategies in the simplex.

## Extensive-Form Correlated Equilibria

Extensive-form correlated equilibria (EFCE) is a natural extension of CE to EFGs. Unlike regular CEs, players do not receive recommendations for the full game upfront; instead, recommendations are received sequentially, and only for infosets the players are currently in. In the original paper by Von Stengel and Forges (2008), this is achieved by means

<sup>1</sup>Perfect recall means there is only one such series of actions.

of *sealed recommendations*, while Farina et al. (2019a) have the mediator generating recommendations over the course of the game, but ceasing all future recommendations if a player deviates from a recommendation. We call the recommended actions *trigger sequences*  $\sigma^1$  (Dudík and Gordon 2009). Trigger sequences contain the last recommended action from the mediator before any deviation, and implicitly contains information about all previous recommendations (due to perfect recall). EFCEs are *incentive-compatible*, players do not expect to benefit by unilaterally deviating.

**Polytope of Correlation Plans** A significant benefit of EFCEs over regular CEs is computational cost: computing a CE that achieves maximum social welfare is NP-complete (Von Stengel and Forges 2008), while in 2-player perfect recall games without chance<sup>2</sup>, the constraints that define an EFCE may be expressed in a polynomial number of linear constraints and hence may be solved using a linear program. Crucial to these positive results is a theorem by Von Stengel and Forges which characterizes  $\Xi$ , the *polytope of correlation plans* which compactly represents the space of joint (reduced) normal-form strategies up to strategic equivalence.

**Definition 1.** (Connected infosets,  $I \Rightarrow I'$ ) Let  $I, I'$  be infosets from either player. We say that  $I, I'$  are connected and write  $I \Rightarrow I'$  if there exists nodes  $u \in I, v \in I'$  in  $\mathcal{G}$  lying on a path starting from the root.

**Definition 2.** (Relevant sequences,  $\sigma_1 \bowtie \sigma_2$ ) Let  $\sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2$ . We say that the sequence pair  $(\sigma_1, \sigma_2)$  is relevant, denoted by  $\sigma_1 \bowtie \sigma_2$  if (i) either  $\sigma_1$  or  $\sigma_2$  is  $\emptyset$  or (ii)  $\sigma_1 = (I_1, a_1), \sigma_2 = (I_2, a_2)$  for  $I_1 \Rightarrow I_2$  and some actions  $a_1, a_2$ . For convenience, we use the same notation  $\sigma_1 \bowtie I_2$  when either  $\sigma_1 = \emptyset$  or if  $\sigma_1 = (I_1, a_1)$  and  $I_1 \Rightarrow I_2$ , with a symmetric definition for  $I_1 \bowtie \sigma_2$ .

**Definition 3.** (Von Stengel and Forges) Let  $\mathcal{G}$  be a perfect recall game without chance. Then,  $\Xi$  is a convex polytope of correlation plans which contains non-negative vectors indexed by relevant sequence pairs, with constraints

$$\Xi := \left\{ \boldsymbol{\xi} \geq 0 : \begin{array}{l} \boldsymbol{\xi}[\emptyset, \emptyset] = 1, \\ \sum_{a \in \mathcal{A}(I)} \xi[(I, a), \sigma_2] = \xi[\sigma(I_1), \sigma_2], \\ \sum_{a \in \mathcal{A}(I)} \xi[\sigma_1, (I_2, a)] = \xi[\sigma_1, \sigma(I_2)] \end{array} \right\}$$

where the second (and third) constraint is over all  $I_1 \bowtie \sigma_2$  ( $\sigma_1 \Rightarrow I_2$ ).

Visually, one can view  $\Xi$  as a 2-dimensional ‘checkerboard’ of size  $|\Sigma_1| \cdot |\Sigma_2|$  with entries to be filled in indices where  $\sigma_1 \bowtie \sigma_2$ . The second and third constraints are simply the sequence-form constraints (Von Stengel and Forges 2008) applied to each row and column of the checkerboard. For example, for the game in Figure 1, all sequence pairs are relevant, and we have row constraints  $\xi[\sigma_1, \ell_x] + \xi[\sigma_1, r_x] = \xi[\sigma_1, \emptyset]$  and  $\xi[\sigma_1, \ell_y] + \xi[\sigma_1, r_y] = \xi[\sigma_1, \emptyset]$  for all sequences  $\sigma_1 \in \Sigma_1$ , and column constraints  $\xi[G, \sigma_2] + \xi[B, \sigma_2] = \xi[\emptyset, \sigma_2]$ ,  $\xi[X_G, \sigma_2] + \xi[Y_G, \sigma_2] = \xi[G, \sigma_2]$ , and  $\xi[X_B, \sigma_2] + \xi[Y_B, \sigma_2] = \xi[B, \sigma_2]$  for all  $\sigma_2 \in \Sigma_2$ .

<sup>2</sup>and more generally in games that are triangle-free(Farina and Sandholm 2020)

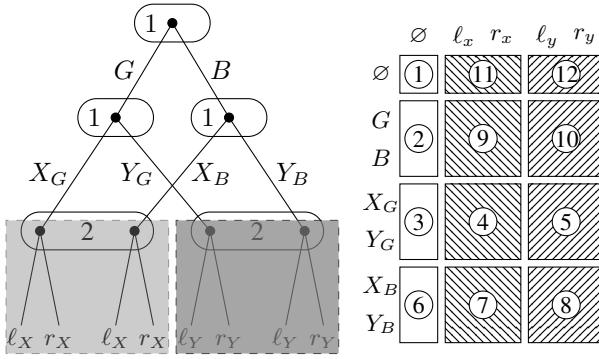


Figure 1: Left: Modified signaling game used in (Von Stengel and Forges 2008) with 2 subgames. Right: Correlation plan  $\xi$ . Circles denote fill-in order under the decomposition of Farina et al. (2019b). Dashed rectangles show sequence pairs in different subgames.

**LP-based EFCE solvers.** Observe that  $\Xi$  contains a polynomial number of unknowns and linear constraints. A correlation plan in  $\Xi$  is a EFCE if it also satisfies incentive constraints that enforce incentive compatibility such that it is optimal for a player to follow the recommendation. Von Stengel and Forges show that the incentive constraints can be also expressed in a polynomial number of linear constraints over  $\xi$ . Specifically, incentive constraints when  $\sigma^! = (I, a^!)$  is recommended for  $P_1$  (the case for  $P_2$  follows naturally) are expressed by<sup>3</sup>

$$\mu(\sigma^!) \geq \beta(\sigma'; \sigma^!) \quad \sigma' = (I, a'), a' \in \mathcal{A}(I) \setminus \{a^!\} \quad (1)$$

$$\mu(\sigma) = \sum_{\sigma_2; (\sigma, \sigma_2) \in \mathcal{L}} u_1(\sigma, \sigma_2) \xi[\sigma, \sigma_2] + \sum_{\sigma' \succ_1 \sigma} \mu(\sigma') \quad (2)$$

$$\beta(\sigma_1; \sigma^!) = \sum_{(\sigma_1, \sigma_2) \in \mathcal{L}} u_1(\sigma_1, \sigma_2) \xi[\sigma^!, \sigma_2] + \sum_{\substack{I' : \sigma(I') \\ = \sigma_1}} \nu(I'; \sigma^!) \quad (3)$$

$$\nu(I; \sigma^!) \geq \beta(\sigma; \sigma^!) \quad a \in \mathcal{I}(I) \quad (4)$$

Here,  $\mu(\sigma)$  gives the expected utility of  $P_1$  if he abides to this and all following recommendations. Together, (3) and (4) recursively define the values of the best response of  $P_1$  for deviating to  $\sigma'$  given  $\sigma^!$  was recommended. The term  $\xi[\sigma^!, \sigma_2]$  essentially contains the (unnormalized) posterior of  $P_2$ 's sequence given that  $\sigma^!$  was recommended.

**Bilinear Saddle-point Problems and Regret Minimization** More recent work by (Farina et al. 2019a,b) show that the problem of finding an EFCE can be formulated as a bilinear saddle point problem, i.e., an optimization problem of the form  $\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} x^T A y$ . Conceptually, this can be seen a zero-sum game between two entities, (i) a *mediator*, who optimizes  $\xi \in \Xi$ , and (ii) a *deviator*, who selects, for each sequence  $\sigma^! \in \Sigma_i$ , the strategy (for all  $\sigma \succ \sigma^!$ ) that is to be taken after deviating from  $\sigma^!$ , given the mediator's choice of  $\xi$ . Essentially, the mediator tries to increase the value of

<sup>3</sup>Readers familiar with the work of Von Stengel and Forges (2008) will notice that we use a slightly different LP. This is to make our future definition of exploitability more convenient.

$\mu$ , while the deviator seeks to increase  $\nu$ , which makes the inequality in (1) more difficult to achieve. Farina et al. characterize  $\Xi$  in terms of a series of convexity-preserving operations known as *scaled extensions* and provide a regret minimizer for sets constructed via scaled extensions. This construction leads to an efficient EFCE solver that runs in linear space, which we adapt in one of our resolving algorithms.

**Quality of correlation plans** The quality of any correlation plan  $\xi$  is measured by (i) its expected *social welfare*,  $\sum_{(\sigma_1, \sigma_2) \in \mathcal{L}} \xi[\sigma_1, \sigma_2] u(\sigma_1, \sigma_2)$ , where  $u$  is typically the payoff sum  $u_1 + u_2$ , and (ii) the degree to which the  $\xi$  violates the incentive constraints.

**Definition 4.** (Exploitability) Given a trigger sequence  $\sigma^!$  of  $P_1$ , and a strategy  $\xi \in \Xi$ , the value of the best-deviating response to  $\sigma^! = (I, a^!)$  is given by

$$\beta^*(\xi; \sigma^!) = \max_{a \in \mathcal{I} \setminus \{a^!\}} \beta((I, a), \xi; \sigma^!)$$

$$\beta(\sigma, \xi; \sigma^!) = \sum_{\sigma_2; (\sigma, \sigma_2) \in \mathcal{L}} u_1(\sigma, \sigma_2) \xi[\sigma^!, \sigma_2] + \sum_{I; \sigma(I) = \sigma} \nu(I, \xi; \sigma^!)$$

$$\nu(I, \xi; \sigma^!) = \max_{a \in \mathcal{I} \setminus \{a^!\}} \beta((I, a), \xi; \sigma^!)$$

with a similar definition for  $P_2$ . The exploitability of  $\xi$  for a trigger sequence  $\sigma^!$  is given by

$$\delta^*(\xi, \sigma^!) = \beta^*(\xi; \sigma^!) - \mu(\xi, \sigma^!)$$

where  $\mu(\xi, \sigma^!)$  is the value of the  $\sigma^!$  if it and all future recommendations are followed, as defined in (2).

$\beta^*(\xi; \sigma^!)$  is the highest reward a player can get from deviating from the trigger sequence  $\sigma^!$ , while  $\delta^*$  measures the gain from doing so. If  $\delta^*(\xi; \sigma^!) \leq 0$  for all  $\sigma^! \in \mathcal{I}_1 \cup \mathcal{I}_2$ , then  $\xi$  is an EFCE. In LP-based solvers, the social welfare is maximized through the objective function, while exploitability is  $\leq 0$  using linear constraints. In the regret minimization method, exploitability is bounded by the average regret incurred by the solvers, which goes to 0 at a rate of  $1/\sqrt{T}$ . Maximizing social welfare with regret minimizers typically requires performing binary search.

### Subgames

An EFG's tree structure provides a natural means of decomposing the problem of solving a game into smaller subproblems over subtrees. However, in imperfect information games, we will require additional restrictions..

**Definition 5.** (Subgame) Let  $\mathcal{G}$  be an EFG with perfect recall. Let  $H$  be a subset of nodes in  $\mathcal{G}$  and  $\check{\mathcal{G}}_H$  be the subgraph induced by  $H$ . We call  $\check{\mathcal{G}}_H$  a subgame of  $\mathcal{G}$  when: (i) if state  $s \in H$ , then  $s' \sqsupset s$  implies  $s' \in H$ , and (ii) for all information sets  $I \in \mathcal{I}_1 \cup \mathcal{I}_2$ , we have  $H \cap I = I$  or  $\emptyset$ .<sup>4</sup>

**Definition 6.** (Subgame Decomposition) Let  $\mathcal{H} = \{H_j\}$  be sets of vertices of  $\mathcal{G}$ . We call  $\mathcal{H}$  a valid subgame decomposition if (i)  $\mathcal{H}$  contains non-intersecting sets, (ii) each  $H_j \in \mathcal{H}$  induces a valid subgame  $\check{\mathcal{G}}_{H_j}$  ( $\check{\mathcal{G}}_j$  for short).

<sup>4</sup>Alternatively if  $h \in \check{\mathcal{G}}$  and belongs to some info-set  $I$ , then all states  $h' \in I$  are contained in  $\check{\mathcal{G}}$ .

For this paper, we will assume that we are equipped with a valid subgame decomposition  $\mathcal{H}$ , which induces  $J$  disjoint subgames  $\{\tilde{\mathcal{G}}_j\}$ . There are many possible ways to obtain subgame decomposition, but by far the most natural and common one is based on *public information*. In this paper, we make no additional assumptions on subgames apart from those in the definition. We call nodes that are not included in any  $\mathcal{H}$  as *pre-subgame*, with an induced subtree  $\hat{\mathcal{G}}$ . Note that  $\hat{\mathcal{G}}$  obeys property (ii) of a subgame; if some infoset is only partially contained in  $\hat{\mathcal{G}}$ , then it must be partially contained in some subgame, which is disallowed. Consequently, leaves, infosets, and sequences may be likewise partitioned. We denote these sets by  $\check{\mathcal{L}}_j, \hat{\mathcal{L}}, \check{\mathcal{I}}_{i,j}, \hat{\mathcal{I}}_i$ , and  $\check{\Sigma}_{i,j}, \hat{\Sigma}_i$ .

The game in Figure 1 has two subgames, both starting off with  $P_2$  making his move. Here,  $P_2$ 's infosets belong to separate subgames, while  $P_1$ 's infosets all lie in  $\hat{\mathcal{G}}$ . Similarly, all of  $P_2$ 's non-empty sequences lie in a subgame, while all of  $P_1$ 's sequences do not. Another valid subgame decomposition is to have all but the root be in a single subgame.

## Subgame Resolving for EFCE

Subgame resolving exploits the sequential nature of EFGs to refine strategies online. We begin with a *correlation blueprint*, typically a guess or approximate of an EFCE.

**Definition 7.** (Correlation blueprint) A correlation blueprint  $\xi_0 \in \Xi$  for the game  $\mathcal{G}$  is an oracle  $\xi_0[\sigma_1, \sigma_2]$  which can be accessed in constant time for all  $\sigma_1 \bowtie \sigma_2$ .

Note that blueprint strategies  $\xi_0$  may not necessarily be stored explicitly: all we require is that its entries may be accessed efficiently. For example, a blueprint may have players playing independently according to sequence form strategies  $\xi^{(i)}(\sigma)$ , such that  $\xi_0[\sigma_1, \sigma_2] = \xi^{(1)}(\sigma_1) \cdot \xi^{(2)}(\sigma_2)$  (no correlation between players' actions in this special blueprint).

At the beginning of the game, players receive recommendations from the blueprint strategy. Once the game enters a subgame, an equilibrium refinement step is performed *only for that subgame entered*, and recommended actions are instead drawn from that refined correlation plan for the rest of the game. Subgame resolving is an *online* method; instead of solving for the equilibrium upfront, it defers part of its computation to when the game is being played. A generic algorithm is shown in Algorithm 1.

**Refinements of correlation blueprint.** Subgame resolving for EFCEs differs significantly from prior work for zero-sum and Stackelberg games. This is because we are now updating relevant sequence pairs of  $\xi_0$  in the correlation polytope  $\Xi$ , which unlike the space of sequence form strategies, has no obvious hierarchical structure. Fortunately, Definitions 3 and 5 provide enough structure to perform resolving.

**Theorem 1.** (Independence between subgames) Let the set  $S_j$  contain relevant sequences (i)  $\sigma_1, \sigma_2 \in \check{\mathcal{G}}_j$ , or (ii)  $\sigma_1 \in \hat{\mathcal{G}}, \sigma_2 \in \check{\mathcal{G}}_j$ , or (iii)  $\sigma_1 \in \check{\mathcal{G}}_j, \sigma_2 \in \hat{\mathcal{G}}$ . Let  $S_0$  be the set of relevant sequence pairs such that  $\sigma_1, \sigma_2 \in \hat{\mathcal{G}}$ . Then  $\{S_0, \dots, S_J\}$  forms a partition of relevant sequence pairs.

A relevant sequence pair  $(\sigma_1, \sigma_2)$  is *pre-subgame*, written  $(\sigma_1, \sigma_2) \in \hat{\mathcal{G}}$  if  $(\sigma_1, \sigma_2) \in S_0$  and  $(\sigma_1, \sigma_2) \in \check{\mathcal{G}}$  if

Algorithm 1: Subgame Resolving

---

**Input:** EFG, blueprint  $\xi_0$

- 1: **while** game is not over **do**
- 2:   **if** currently in some subgame  $j$  **then**
- 3:     **if** first time in subgame **then**
- 4:       (\*) Refine  $\xi_0 \rightarrow \tilde{\xi}_j$
- 5:     **end if**
- 6:     Recommend action according to  $\tilde{\xi}_j$
- 7:   **else**
- 8:     Recommend action according to  $\xi_0$
- 9:   **end if**
- 10: **end while**

---

$(\sigma_1, \sigma_2) \in S_j$ . Theorem 1 shows exactly one of these must hold.

**Definition 8.** (Refinements) For a given blueprint  $\xi_0 \in \Xi$  and subgame decomposition  $\mathcal{H}$ , a correlation plan  $\tilde{\xi} \in \Xi$  is called a complete refinement if  $\tilde{\xi}[\sigma_1, \sigma_2] = \xi_0[\sigma_1, \sigma_2]$  for all  $(\sigma_1, \sigma_2) \in \hat{\mathcal{G}}$ . Let  $\Xi_j$  be  $\Xi$  but restricted to sequence pairs  $(\sigma_1, \sigma_2) \in \check{\mathcal{G}}_j \cup \hat{\mathcal{G}}$ . We call  $\tilde{\xi}_j \in \Xi_j$  a refinement of subgame  $j$  if  $\tilde{\xi}_j[\sigma_1, \sigma_2] = \xi_0[\sigma_1, \sigma_2]$  for all  $(\sigma_1, \sigma_2) \in \hat{\mathcal{G}}$ .

For example, in Figure 1, a complete refinement involves updating all but the first column, since for  $P_2$ , all but the empty sequence is in some subgame. For the left subgame, we have  $\Xi_j$  being the first 3 columns; finding a refinement involves updating the columns containing  $\ell_x, r_x$  (sequences which are contained in the subgame) and dropping the last 2 columns, while respecting the constraints in Definition 3.

Theorem 1 implies that updated entries (shaded entries) for each refinement do not overlap, hence, refined correlation plans can be combined to form complete refinements. Let  $\{\tilde{\xi}_j\}$  contain a refinement for each subgame. Then,  $\{\tilde{\xi}_j\}$  induces a complete refinement naturally,  $\tilde{\xi}[\sigma_1, \sigma_2] = \xi_0[\sigma_1, \sigma_2]$  if  $(\sigma_1, \sigma_2) \in \hat{\mathcal{G}}$  and  $\tilde{\xi}_j[\sigma_1, \sigma_2]$  if  $(\sigma_1, \sigma_2) \in \check{\mathcal{G}}_j$ . This direct mapping satisfies  $\tilde{\xi} \in \Xi$ , as no constraint of  $\Xi$  involves sequences pairs belonging to different subgames.

The independence property of sequence pairs extends to EFCE incentive constraints for trigger sequences within subgames. For every trigger sequence  $\sigma^!$  in  $\check{\mathcal{G}}_j$ , the best-deviating response (see Definition 4) will never have to reference sequence pairs containing any sequence outside  $\check{\mathcal{G}}_j$ .<sup>5</sup> These show it may be possible to perform refinements of subgames *independently* without solving other entries containing sequences from other subgames. However, independence of incentive constraints does not apply to pre-subgame trigger sequences  $\sigma^! \in \hat{\Sigma}_i$ . For those sequences,  $\delta^*(\xi, \sigma^!)$  will in general depend on refined solutions from multiple, distinct subgames. Handling these constraints is a primary challenge addressed in this paper.

**Safe refining algorithms** An important property when performing subgame-resolving for independent, uncorre-

<sup>5</sup>This is intuitively true. Once inside  $\check{\mathcal{G}}_j$ , a potential deviating player will never encounter states outside of  $\check{\mathcal{G}}_j$  in the future, and hence need not consider them.

lated strategies is that of safety, and was the central issue discussed extensively in solving NE in zero-sum games (Brown and Sandholm 2017). There, it was observed naive application of resolving algorithms can result in solutions which are of lower quality than the blueprint. The fundamental problem is that when  $P_1$  performed resolving, the best-response of  $P_2$  in the pre-subgame portion differs from the blueprint, hence whatever initial distribution over states at the beginning of the subgame no longer holds. This phenomenon is known as *unsafe* resolving. A similar phenomenon quantified in terms of exploitability holds for EFCEs.

**Definition 9.** (Safe refinements) A complete refinement  $\tilde{\xi}$  of  $\xi_0$  is safe if for all trigger sequences  $\sigma^!$

$$\delta^*(\tilde{\xi}; \sigma^!) \leq \max(0, \delta^*(\xi_0; \sigma^!)),$$

i.e., the exploitability of  $\tilde{\xi}$  for all  $\sigma^!$  is 0 or less than the blueprint. We say that  $\tilde{\xi}$  is fully safe if in addition, the social welfare (assuming no deviations) under  $\tilde{\xi}$  is no less than  $\xi_0$ . A resolving algorithm is said to be (fully) safe if the complete refinement induced by all  $j$  refinements  $\tilde{\xi}_j$  is (fully) safe.

In safe refinements, players are at least as incentivized to follow the resolved strategy than the blueprint. Fully safe refinements ensures further that the social welfare will not be diminished. Apart from incurring additional computing costs, there can be no harm in applying fully safe resolving. Clearly, a fully safe resolving algorithm exists in the form of one that trivially returns the blueprint.

**Resolving with multiple subgames.** In Definition 9, we required that the induced complete refinement  $\tilde{\xi}$  be used to measure safety, and not just the refined strategy of a subgame  $\tilde{\xi}_j$ . This may seem odd at first, since the primary advantages of resolving was that it did not require computing strategies for subgames not reached in actual play. However, it turns out that this is necessary. Consider the perspective of  $P_i$  who in the pre-subgame portion of  $\mathcal{G}$  was recommended a sequence  $\sigma^!$  and was considering deviation. At that point of decision making,  $P_i$  does not know which subgame will be reached in the future; however, he knows that whichever subgame is encountered (if at all), refinement will be performed. Thus, when contemplating deviation,  $P_i$  in fact computes the value of a best-deviating response to the complete refinement  $\tilde{\xi}$ . This is despite the fact that in a single playthrough of the game, at most one subgame can be encountered in reality. Another interpretation is that the mediator publishes the refinement *algorithm* which implicitly defines the complete  $\tilde{\xi}$ , which players contemplate best responses to. Hence, even though the resolving algorithm does not explicitly compute a complete refinement, it should still guarantee safety as if it did.

## Safe Subgame Resolving Using LPs

Suppose the mediator has thus far given recommendations based on  $\xi_0$  and the players have just entered subgame  $j$ . Following Algorithm 1, the mediator computes a refinement  $\tilde{\xi}_j$  which he uses for all future recommendations. We now

present a safe refinement algorithm using a LP (a fully safe variant will be discussed later).

On top of the structural constraints of  $\Xi_j$ , we have 3 categories (A-C) of additional constraints that ensure safety. Constraint set (A) enforces safety for trigger sequences  $\sigma^! \in \tilde{\mathcal{G}}_j$ , in an manner identical to (1), while constraint sets (B-C) ensures that the complete refinement  $\tilde{\xi}$  is safe; loosely speaking, (B) contains lower bounds that ensure that following recommendations will yield a high enough payoff to a player contemplating deviation, while (C) contains upper bounds which ensure that players which have deviated do not get rewarded too much.

**(A) Safety for in-subgame triggers** For each  $P_i$  and each sequence in subgame  $j$ , i.e.,  $\sigma^! = (I^!, a^!) \in \tilde{\Sigma}_{i,j}$ , we require

$$\mu(\tilde{\xi}; \sigma^!) \geq \beta^*(\tilde{\xi}; \sigma^!) - \delta^*(\xi_0; \sigma^!) \quad (5)$$

where the  $\mu, \nu$  have constraints identical to (2), (4). These constraints only involve sequence (pairs) that lie within  $\tilde{\mathcal{G}}_j$  and not other subgames, so no modifications are needed.

**Computing safe infoset value bounds** Now we turn to constraints (B) and (C), which guarantee safety for trigger sequences in  $\hat{\mathcal{G}}$ . Our approach is to, for each trigger-sequence  $\sigma^! = (I, a^!)$ , generate a set of linear constraints which guarantee that the safety for  $\sigma^!$  is satisfied, in accordance to Definition 9. What are some sufficient conditions on  $\mu(\tilde{\xi}; \sigma^!)$  and  $\beta(\sigma', \tilde{\xi}; \sigma^!)$  such that the safety condition in Definition 9 is satisfied for  $\sigma^!$ ? To answer this, let us consider  $\alpha = \max(0, \delta^*(\xi_0; \sigma^!))$ . There are 2 cases. (i) If  $\alpha = 0$ , then the blueprint was already sufficiently unexploitable for  $\sigma^!$ . Thus we could afford to decrease  $\mu(\tilde{\xi}; \sigma^!)$  and increase  $\beta(\sigma', \tilde{\xi}; \sigma^!)$  relative to the blueprint—if it leads to better social welfare. (ii) If  $\alpha \geq 0$ , then  $\sigma^!$  was exploitable and we do not want to worsen exploitability. This can be avoided if we could somehow ensure  $\mu(\tilde{\xi}; \sigma^!)$  and  $\beta(\sigma', \tilde{\xi}; \sigma^!)$  do not decrease or increase respectively. Concretely, in case (i), we can require  $\beta^*(\tilde{\xi}; \sigma^!) \leq \hat{\beta}(\sigma; \sigma^!) = \beta^*(\xi_0; \sigma^!) - \delta^*(\xi_0; \sigma^!)/2$ , and  $\mu(\tilde{\xi}; \sigma^!) \geq \check{\mu}(\sigma; \sigma^!) = \mu(\xi_0, \sigma^!) + \delta^*(\xi_0; \sigma^!)/2$ . In case (ii), we can require  $\beta^*(\tilde{\xi}; \sigma^!) \leq \hat{\beta}(\sigma; \sigma^!) = \beta^*(\xi_0; \sigma^!)$ , and  $\mu(\tilde{\xi}; \sigma^!) \geq \check{\mu}(\sigma; \sigma^!) = \mu(\xi_0, \sigma^!)$ . These are sufficient conditions to guarantee that safety is maintained for  $\sigma^!$ . Yet, enforcing this is not possible, since  $\mu(\tilde{\xi}; \sigma^!)$  and  $\beta(\sigma', \tilde{\xi}; \sigma^!)$  can depend on relevant sequence pairs belonging to other subgames. The trick is to recursively unroll  $\mu$  and  $\beta$ , maintaining bounds which guarantee for safety at each step. This is repeated until we reach infosets belonging to subgames.

**Definition 10.** (Head infosets) For a subgame  $j$ ,  $I \in \mathcal{I}_i$  is a head infoset of subgame  $j$  if  $I \in \hat{\mathcal{I}}_{i,j}$  and there does not exist  $I' \prec I$  such that  $I' \notin \hat{\mathcal{I}}_i$ . The set of head infosets for player  $i$  in subgame  $j$  is denoted by  $\mathcal{I}_{i,j}^h \subseteq \check{\mathcal{I}}_{i,j}$ .  $I$  is called a head infoset if it is a head infoset of some subgame.

**(B) Lower bounds on  $\mu(\tilde{\xi}, \sigma^!)$**  Recall that  $\mu(\tilde{\xi}, \sigma^!)$  is the expected utility accrued from leaves  $(\sigma_1, \sigma_2) \in \mathcal{L}$ , where  $\sigma_1 \succeq \sigma^!$ . We can recursively decompose  $\mu(\tilde{\xi}, \sigma^!)$  into values

of infosets, sequences and their summations.

$$d(\sigma; \sigma^!) = (\mu(\xi_0, \sigma) - \check{\mu}(\sigma; \sigma^!)) / |\{I | \sigma(I) = \sigma\}| \quad (6)$$

$$\check{v}(I; \sigma^!) = v(I) - d(\sigma(I); \sigma^!) \quad (7)$$

$$f(I; \sigma^!) = (v(\xi_0, I) - \check{v}(I; \sigma^!)) / |\mathcal{A}(I)| \quad (8)$$

$$\check{\mu}(\sigma; \sigma^!) = \mu(\xi_0, \sigma) - f(I; \sigma^!) \quad I : \sigma = (I, a) \quad (9)$$

where  $v(\xi_0, I) = \sum_{\sigma:(I,a), a \in \mathcal{A}(I)} \mu(\xi_0, \sigma)$ . For every  $\sigma$ , starting from  $\sigma^!$ , we compute in (6) the *slack*, i.e., the difference between our desired lower bound  $\check{\mu}(\sigma)$  and what was achieved with the blueprint. In (7), this slack is split equally between all infosets which have  $\sigma$  as the parent sequence. A similar process is repeated for infosets in (8) and (9). We alternate between computing lower bounds for sequences and infosets until we have computed  $\check{v}(I)$  for  $I \in \mathcal{I}_i^{\text{head}}$  in (7). We repeat this for all  $\sigma^! \in \hat{\mathcal{G}}$  and take the tighter of the bounds to obtain  $\check{v}(I^{\text{head}})$  for all  $I^{\text{head}} \in \mathcal{I}_i^{\text{head}}$ .

**(C) Upper bounds on  $\beta^*(\tilde{\xi}; \sigma^!)$**  Recall that  $\beta^*$  is the value of the best-deviating response. We can unroll the inequalities using Definition 4 and stop once a head infoset is reached, i.e., when we encounter a term  $\nu(I, \tilde{\xi}; \sigma^!)$  for some  $I \in \mathcal{I}_i^{\text{head}}$ . If these terms were upper-bounded appropriately, then  $\beta^*(\tilde{\xi}; \sigma^!)$  would be upper-bounded. One possible way is to compute upper bounds recursively

$$s(\sigma; \sigma^!) = (\hat{\beta}(\sigma; \sigma^!) - \beta(\sigma, \xi_0; \sigma^!)) / |\{I | \sigma(I) = \sigma\}| \quad (10)$$

$$\hat{\nu}(I; \sigma^!) = \nu(I, \xi_0; \sigma^!) + s \quad (11)$$

$$\hat{\beta}(\sigma; \sigma^!) = \hat{\nu}(I; \sigma^!). \quad (12)$$

At the end of the bounds computation step, we have sets  $\hat{\mathcal{B}}_{i,j} = \{(I, \sigma^!, \hat{\nu}(I; \sigma^!)\}$  and  $\check{\mathcal{B}}_{i,j} = \{(I, \check{v}(I))\}$ , containing constraints of the form  $v(\tilde{\xi}; I^{\text{head}}) \geq \check{v}(I^{\text{head}})$  and  $\nu(I^{\text{head}}, \tilde{\xi}; \sigma^!) \leq \hat{\nu}(I; \sigma^!)$  for some  $I \in \mathcal{I}_{i,j}^{\text{head}}$ .

**Theorem 2.** *If  $\tilde{\xi}$  satisfies all constraints in  $\check{\mathcal{B}}_{i,j}$  and  $\hat{\mathcal{B}}_{i,j}$  for all  $i \in \{1, 2\}$  and  $j \in [J]$ , then  $\delta^*(\tilde{\xi}; \sigma^!) \leq \max(0, \delta^*(\xi_0; \sigma^!))$  for all  $\sigma^! \in \hat{\mathcal{G}}$ .*

**Piecing the LP together** Once these bounds are computed, enforcing them is simply a matter of placing them on top of the constraints for trigger sequences in  $\check{\mathcal{G}}_j$  in (5). For lower bounds of the form  $(I, (I)) \in \check{\mathcal{B}}_{i,j}$ , we introduce variables  $v(I)$ , where  $v(I) = \sum_{\sigma:(I,a), a \in \mathcal{A}(I)} \mu(\sigma)$  and enforce  $v(I) \geq \check{v}(I)$ . Note that the auxiliary variables  $\mu(\sigma)$  has already been introduced as part of exploitability of  $\check{\mathcal{G}}_j$  when enforcing (5). For upper bounds  $(I, \sigma^!, \hat{\nu}(I; \sigma^!)) \in \hat{\mathcal{B}}_{i,j}$ , we introduce variables  $\nu(I; \sigma^!)$  and enforce  $\nu(I; \sigma^!) \leq \hat{\nu}(I; \sigma^!)$ . To ensure that  $\nu(I; \sigma^!)$  is indeed the value of the infoset given a trigger sequence  $\sigma^!$ , we will have to introduce auxiliary variables similar to (3), (4) recursively. A summary and more precise explanation is included in the appendix. This LP is always feasible, since the blueprint would trivially satisfy all bounds constraints. To achieve full safety, we simply set the objective to be the component of social welfare culminating from subgame  $j$ .

## Subgame Resolving with Regret Minimization

Our second algorithm is based on regret minimization. We solve a saddle-point problem using self-play, utilizing the scaled extension operator of Farina et al. (2019b) to provide an efficient regret minimizer over  $\Xi_j$ . This leads to a significantly more efficient algorithm.

**Refinements as a Bilinear Saddle-point Problem** First, we show that the refinement LP may be written as a bilinear saddle point problem, similar to what was done in Farina et al. (2019a). Observe that a refinement  $\tilde{\xi}_j$  is safe if and only if the greatest violation of the safety constraints to be equal to 0. Building on this intuition, we introduce for each safety constraint, multipliers  $\lambda_{i,\sigma^!}^\delta, \lambda_{i,I,\sigma^!}^\nu$  and  $\lambda_{i,I}^v$  — for exploitability (in  $\check{\mathcal{G}}_j$ ), upper bounds, and lower bounds respectively. These multipliers are non-negative and sum to 1. Additionally, we introduce variables  $\check{y}_{i,\sigma^!} \in \check{Y}_{i,\sigma^!}$  for  $(I^!, a^!) = \sigma^! \in \check{\Sigma}_{i,j}$ . Similarly, for trigger sequences  $(I^!, a^!) = \sigma^! \in \hat{\Sigma}_{i,j}$ , we introduce  $\hat{y}_{i,\sigma^!} \in \hat{Y}_{i,\sigma^!}$ . These  $y$ 's represent the components of the best-deviating responses to trigger sequences  $\sigma^!$ , and whose polytopes can be easily represented using the sequence-form representation of Von Stengel (1996). We explain in more detail in the Appendix. Resolving is equivalent to solving the following bilinear saddle point problem:

$$\min_{\tilde{\xi}_j} \max_{i, \lambda, y} \left\{ \begin{array}{l} \sum_{i, \sigma^! \in \check{\Sigma}_{i,j}} \left[ \tilde{\xi}_j^T R^\delta z_{i,\sigma^!}^\delta + \tilde{\xi}_j^T \left( \lambda_{i,\sigma^!}^\delta b_{i,\sigma^!}^\delta \right) \right] + \\ \sum_{i, (I, \sigma^!, \cdot) \in \hat{\mathcal{B}}_{i,j}} \left[ \tilde{\xi}_j^T R^\nu z_{i,\sigma^!}^\nu + \tilde{\xi}_j^T \left( \lambda_{i,\sigma^!}^\nu b_{i,\sigma^!}^\nu \right) \right] + \\ \sum_{i, (I, \cdot) \in \check{\mathcal{B}}_j} \tilde{\xi}_j^T \left( \lambda_{i,I}^v b_{i,I}^v \right) \end{array} \right\}, \quad (13)$$

where  $z_{i,\sigma^!}^\delta = \lambda_{i,\sigma^!}^\delta \check{y}_{i,\sigma^!}$  and  $z_{i,\sigma^!}^\nu = \lambda_{i,\sigma^!}^\nu \hat{y}_{i,\sigma^!}$ , for appropriately chosen constants  $R, b$  (which may vary on  $\xi_0$ ). Hence, we can treat the refinement problem as a *zero-sum* game between a *mediator*, who chooses a refinement  $\tilde{\xi}_j$  and *deviator*, who chooses multipliers and best-deviating responses. This zero-sum game can be solved by running self-play between two Hannan-consistent regret minimizers and taking average strategies. A regret minimizer for the deviator can be constructed efficiently using counterfactual regret minimization (Zinkevich et al. 2007). A regret minimizer over  $\Xi_j$  is constructed using the decomposition technique used by Farina et al. (2019b) with some additional tiebreaking rules to ensure we do not have to "fill-in" sequence pairs in  $\hat{\mathcal{G}}$ . The algorithm is outlined in Algorithm 2, with the full details of the modified decomposition algorithm and deviator polytope presented in the appendix.

## Experiments

We evaluate our algorithms using the LP-based and regret minimization-based refining. We use the benchmark game of EFCE called *Battleship*, introduced by Farina et al. (2019a). This game is played in 2 stages. In the *placement* stage, players privately place their ship(s) of size 1 by  $m$  on  $W \times H$  grid. In the *firing* stage, players take turns firing at each other

---

**Algorithm 2: Refinement with Regret Minimization**


---

**Input:** EFG, blueprint  $\xi_0$ 

- 1: Decompose  $\Xi_j$  into series of scaled extensions.
  - 2: Construct regret RM'er  $\mathcal{X}$  over  $\Xi_j$ .
  - 3: Construct regret RM'er  $\mathcal{Y}$  over deviators.
  - 4: **while** saddle point gap  $\geq \epsilon$  **do**
  - 5:    $\xi_j^{(t)} \leftarrow \mathcal{X}.\text{recommend}; y^{(t)} \leftarrow \mathcal{Y}.\text{recommend}$
  - 6:    $\mathcal{X}.\text{observeLoss}(y_t); \mathcal{Y}.\text{observeLoss}(\xi_t)$
  - 7: **end while**
- 

over  $T$  timesteps, or until a player's ship is destroyed. Each shot is at a single tile, and a ship is considered destroyed when all tiles in the ship are shot at least once. A player gets 1 point for destroying the opponent's ship, but loses  $\gamma$  points if his ship is destroyed. If no ship is destroyed by the end of the game, the game ends in a tie and both players get 0.

We use 2 different correlation blueprints for our experiments, *Uniform* and *Jittered*. Both correlation plans are based on independent player strategies stored using the sequence form. That is,  $\xi_0[\sigma_1, \sigma_2] = \xi_0^{(1)}(\sigma_1) \cdot \xi_0^{(2)}(\sigma_2)$ , where  $\xi_0^{(1)}, \xi_0^{(2)}$  are sequence form strategies for each player. In *Uniform*,  $\xi_0^{(i)}$  have actions uniformly at random at each info-set. In *Jittered*, each player has randomly generated behavioral strategies. Here, for info-set  $I \in \mathcal{I}_i$ , action  $a_j \in \mathcal{A}(I)$  is played with probability  $p(a_j; I) = \kappa_{I,j} / \sum_k \kappa_{I,k}$ , with  $\kappa_{I,k} = 1 + w \cdot \varepsilon_{I,k}$ , where each  $\varepsilon_{I,k}$  is drawn independently and uniformly from  $[-1, 1]$  and  $w \in [0, 1]$  is a width parameter governing the level of deviation from uniform strategies.

Subgames are defined based on public information, which at the  $k$ -th step of firing are precisely the locations fired by each player. We base subgames on the shot history up till timestep  $T' < T$ .  $T'$  balances the trade-off between accuracy versus computational costs. For a grid of size  $n$ , we have  $J = \prod_{k=T-T'+1}^T k^2$  subgames. When  $T'$  is small, we have fewer subgames, but can achieve better social welfare. All experiments are run on an Apple M1 Chip with 16GB of RAM with 8 cores. LPs are solved using Gurobi (Gurobi Optimization, LLC 2021).

**Safe resolving with SW maximization** We first show using our LP-based method that ensures fully safe resolving can lead to significantly higher social welfare as compared to the blueprint. We set  $T' = 1$  and we use ships with  $m = 1$ , i.e., the game is over once any ship is hit. Consequently, the game is entirely symmetric in terms of location. The NE here is to play and shoot uniformly at random. Hence, *Uniform* is a valid, though not SW-optimal EFCE. Under *Uniform*, the exploitability  $\delta^*$  under the blueprint is 0, implying that the complete refinement  $\tilde{\xi}$  is also an EFCE. We perform refinement on the first subgame (this without loss of generality due to symmetry) and compare the SW accumulated from the subgame under the blueprint and refinement. For *Jittered*, we repeated the experiment 10 times with different seeds and report the mean. The results are reported in Table 1. In all our experiments, our refined strategy  $\xi_j$  gives a much higher SW. For example, in the largest example with  $\gamma = 2$ ,

$n, T$	$J$	$ \Xi_j $	$\gamma$	Uniform		Jittered	
				BP	Refined	BP	Refined
3, 2, 9	382	2 5	-3.70 -14.8	-3.70	-14.8	-3.55	-3.55
				-14.8	-14.8	-14.2	-14.2
4, 3, 16	3.2e3 2.3e4	2 5	-3.13 -12.5	-2.95	-3.24	-3.10	-11.8
				-12.5	-11.4	-13.0	-11.8
5, 3, 25	2.3e4 1.2e5	2 5	-1.92 -7.68	-1.34	-1.95	-1.25	-4.32
				-7.68	-4.80	-7.82	-4.32
6, 3, 36	1.2e5 36	2 5	-1.23 -4.94	-.772	-1.25	-.627	
				-4.94	-2.47	-4.99	-1.95

Table 1: Comparison of social welfare between blueprint (BP) and SW-maximizing safe refinement with ships of size 1. Social welfare is reported at a scale of 1e-2.

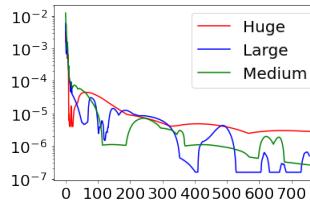


Figure 2: Left: Most violated incentive constraint of  $\tilde{\xi}$  plot against iteration number. Right: Parameters of game.

SW increases by 4.6e-3. This is not a negligible improvement; since this is applied to all 36 subgames, the expected improvement in SW of the complete refinement  $\tilde{\xi}$  is actually 0.167.  $|\Xi_j|$  is significantly smaller than  $|\Xi|$ , such that each refinement is computed in no more than 10 seconds.

**Safe resolving using regret minimization** We now demonstrate the scalability of refinement based on regret minimization. Our goal here is to demonstrate that subgame resolving can be performed efficiently for games that are too large for  $\xi$  to even be stored in memory. We run refinement using our regret minimization algorithm and report the "pseudo"-exploitability of  $\tilde{\xi}_j$  (i.e., the value of the inner maximization over  $(i, \lambda, y)$ , (13), or the most violated incentive constraint of the LP). We use  $T' = 1, \gamma = 2$  and the *Uniform* blueprint. The results are reported in Figure 2. Our huge instance is several times larger than the largest instance in Farina et al. (2019b), and it would require a significant amount of memory to store a full correlation plan  $\tilde{\xi}$ . We find that in practice, resolving requires less than 0.5 seconds per iteration, while using no more than 2GB of memory.

## Conclusion

In this paper, we propose a novel subgame resolving technique for EFCE. We offer two algorithms, the first based on LPs and the second uses regret minimization, both of which consume significantly less compute than full-game solvers. Our technique is, to the best of our knowledge, the first *online* algorithm towards solving EFCE. In future, we hope to expand our work to other equilibria, such as those involving hindsight rationality (Morrill et al. 2021).

## Acknowledgements

This work was supported in part by NSF grant IIS-2046640 (CAREER) and a research grant from Lockheed Martin.

## References

- Brown, N.; and Sandholm, T. 2017. Safe and nested endgame solving for imperfect-information games. In *Workshops at the thirty-first AAAI conference on artificial intelligence*.
- Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374): 418–424.
- Dudík, M.; and Gordon, G. J. 2009. A Sampling-Based Approach to Computing Equilibria in Succinct Extensive-Form Games. In *UAI*.
- Farina, G.; Kroer, C.; and Sandholm, T. 2019. Regret circuits: Composability of regret minimizers. In *International conference on machine learning*, 1863–1872. PMLR.
- Farina, G.; Ling, C. K.; Fang, F.; and Sandholm, T. 2019a. Correlation in Extensive-Form Games: Saddle-Point Formulation and Benchmarks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Farina, G.; Ling, C. K.; Fang, F.; and Sandholm, T. 2019b. Efficient Regret Minimization Algorithm for Extensive-Form Correlated Equilibrium. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Farina, G.; and Sandholm, T. 2020. Polynomial-time computation of optimal correlated equilibria in two-player extensive-form games with public chance moves and beyond. *arXiv preprint arXiv:2009.04336*.
- Gray, J.; Lerer, A.; Bakhtin, A.; and Brown, N. 2021. Human-Level Performance in No-Press Diplomacy via Equilibrium Search. In *International Conference on Learning Representations*.
- Gurobi Optimization, LLC. 2021. Gurobi Optimizer Reference Manual.
- Lerer, A.; Hu, H.; Foerster, J.; and Brown, N. 2020. Improving policies via search in cooperative partially observable games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 7187–7194.
- Ling, C. K.; and Brown, N. 2021. Safe Search for Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 5541–5548.
- Morrill, D.; D’Orazio, R.; Lanctot, M.; Wright, J. R.; Bowling, M.; and Greenwald, A. 2021. Efficient Deviation Types and Learning for Hindsight Rationality in Extensive-Form Games. *arXiv preprint arXiv:2102.06973*.
- Von Stengel, B. 1996. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2): 220–246.
- Von Stengel, B.; and Forges, F. 2008. Extensive-form correlated equilibrium: Definition and computational complexity. *Mathematics of Operations Research*, 33(4): 1002–1022.
- Zhang, B. H.; and Sandholm, T. 2021. Subgame solving without common knowledge. *arXiv preprint arXiv:2106.06068*.
- Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. *Advances in neural information processing systems*, 20: 1729–1736.