Faster algorithms for extensive-form game solving via improved smoothing functions

Christian Kroer*
Computer Science Department
Carnegie Mellon University
ckroer@cs.cmu.edu

Kevin Waugh
Department of Computing Science
University of Alberta
kevin.waugh@gmail.com

Fatma Kılınç-Karzan†
Tepper School of Business
Carnegie Mellon University
fkilinc@andrew.cmu.edu

Tuomas Sandholm‡
Computer Science Department
Carnegie Mellon University
sandholm@cs.cmu.edu

September 24, 2018

Abstract

Sparse iterative methods, in particular first-order methods, are known to be among the most effective in solving large-scale two-player zero-sum extensive-form games. The convergence rates of these methods depend heavily on the properties of the distance-generating function that they are based on. We investigate both the theoretical and practical performance improvement of first-order methods (FOMs) for solving extensive-form games through better design of the dilated entropy function—a class of distance-generating functions related to the domains associated with the extensive-form games. By introducing a new weighting scheme for the dilated entropy function, we develop the first distance-generating function for the strategy spaces of sequential games that has only a logarithmic dependence on the branching factor of the player. This result improves the overall convergence rate of several first-order methods working with dilated entropy function by a factor of $\Omega(b^d d)$, where b is the branching factor of the player, and d is the depth of the game tree.

Thus far, counterfactual regret minimization methods have been faster in practice, and more popular, than first-order methods despite their theoretically inferior convergence rates. Using our new weighting scheme and a practical parameter tuning procedure we show that, for the first time, the excessive gap technique, a classical first-order method, can be made faster than the counterfactual regret minimization algorithm in practice for large games, and that the aggressive stepsize scheme of CFR+ is the only reason that the algorithm is faster in practice.

1 Introduction

Extensive-form games (EFGs) are a broad class of games; they model sequential interaction, imperfect information, and outcome uncertainty. Nash equilibria prescribe a particular notion of rational behavior in such games. In the specific case of two-player zero-sum EFGs with perfect recall, an exact Nash equilibrium

*Supported by a Facebook Fellowship, the NSF under grants IIS-1617590, IIS-1320620, and IIS-1546752, and the ARO under awards W911NF-16-1-0061 and W911NF-17-1-0082.

†Supported by NSF grant CMMI 1454548.

‡Supported by the NSF under grants IIS-1617590, IIS-1320620, and IIS-1546752, and the ARO under awards W911NF-16-1-0061 and W911NF-17-1-0082.

can be computed in polynomial time using a Linear Program (LP) whose size is linear in the size of the game tree [37]. However, in practice the LP approach has two major drawbacks limiting its applicability. First, the LP may be prohibitively large and may not fit in memory. Second, even when it does, the iterations of interior-point methods or the simplex algorithm are prohibitively expensive [33]. Practical methods for EFG solving tackle this issue through two complementary approaches: Abstraction and iterative game solvers with low memory requirements [33]. In this paper we focus on the second approach. Iterative game solvers mainly fall in two categories: (i) counterfactual-regret-based methods [23, 39] achieving a convergence rate on the order of $O(\frac{1}{\varepsilon^2})$, and (ii) first-order methods (FOMs) [12, 20] achieving a convergence rate of $O(\frac{1}{\varepsilon})$. The better convergence rate of FOMs has made them more attractive from a theoretical viewpoint, yet so far these techniques have been outperformed by counterfactual-regret based methods in practice. This paper investigates the performance improvements of FOMs for EFGs, from both a theoretical and a numerical perspective. Through our new theoretical and practical developments, we establish, for the first time, that FOMs can be made state-of-the-art in practice as well.

Nash equilibrium computation of a two-player zero-sum EFG with perfect recall admits a Bilinear Saddle Point Problem (BSPP) formulation where the domains are given by the polytopes that encode strategy spaces of the players. There are a number of efficient and well-known FOMs designed to solve BSPPs. The classical FOMs to solve BSPPs such as *mirror prox* (MP) [28] or the *excessive gap technique* (EGT) [29] utilize *distance-generating functions* (DGFs) to measure appropriate notions of distances over the domains. Consequently, the convergence rate of these FOMs relies on the DGFs and their relation to the domains in three critical ways: Through the strong convexity parameters of the DGFs, the norm associated with the strong convexity parameter, and the set widths of the domains as measured by the DGFs.

Hoda et al. [12] introduced a general framework for constructing DGFs for *treeplexes*—a class of convex polytopes that generalize the domains associated with the strategy spaces of an EFG. While they also established lower bounds on the strong convexity parameter for their DGFs in some special cases, these lead to very weak bounds and result in slow convergence rates. Kroer et al. [20] developed explicit strong convexity-parameter bounds for entropy-based DGFs (a particular subclass of DGFs) for general EFGs, and improved the bounds for the special cases considered by Hoda et al. [12]. These bounds from Kroer et al. [20] generate the current state-of-the-art parameters associated with the convergence rate for FOMs with $O(\frac{1}{\varepsilon})$ convergence.

In this paper we construct a new weighting scheme for such entropy-based DGFs. This weighting scheme leads to new and improved bounds on the strong convexity parameter associated with general treeplex domains. In particular, our new bounds are first-of-their kind as they have only a logarithmic dependence on the branching operation of the treeplex. Informally, our strong convexity result allows us to improve the convergence rate of FOMs working with entropy-based DGFs by a factor of $\Omega(b^d d)$ (where b is the average branching factor for a player and d is the depth of the EFG) compared to the prior state-of-the-art results from Kroer et al. [20]. In terms of their logarithmic dependence on the branching factor, our bounds parallel the simplex case for matrix games where the entropy function achieves a logarithmic dependence on the dimension of the simplex domain.

Finally, we complement our theoretical results with numerical experiments to investigate the speed up of FOMs with convergence rate $O(\frac{1}{\varepsilon})$ and compare the performance of these algorithms with the premier regret-based methods CFR and CFR+ [36], which have a theoretical convergence rate of $O(\frac{1}{\varepsilon^2})$. CFR+ is the fastest prior algorithm for computing Nash equilibria in EFGs when the entire tree can be traversed (rather than sampled). Bowling et al. [3] used it to essentially solve the game limit Texas hold'em. CFR+ is also the algorithm used to accurately solve endgames in the Libratus agent, which showed superhuman performance against a team of top Heads-Up No-Limit Texas hold'em poker specialist professional players in the Brains

vs AI event [1]. A slight variation[2] of CFR+ was used in the DeepStack agent Moravčík et al. [27], which beat a group of professional players.

We perform numerical experiments on scaled-up variants of *Leduc hold'em* [35], a poker game that has become a standard benchmark in the EFG-solving community, as well as a security-inspired attacker/defender game played on a graph. The performance we get from our FOM-based approach with EGT relative to CFR and CFR+ is in sharp contrast to the previous conventional practical wisdom in the field. Previously it was thought that FOM-based methods converged faster than CFR ultimately, but that CFR had a faster initial convergence and the cross-over point occurred later as games got larger, see e.g. Kroer et al. [20]. Our experiments show that FOMs are substantially faster than CFR algorithms when using a practically-tuned variant of our DGF, even as the game-size is scaled up and when CFR is using the RM+ regret minimizer. We find that CFR+ is still faster in practice, but only due to its aggressive stepsize policy. This suggests that future work on FOMs could be coupled with our DGF to create state-of-the-art algorithms in practice. We also test the impact of stronger bounds on the strong convexity parameter: we instantiate EGT with the parameters developed in this paper, and compare the performance to the parameters developed by Kroer et al. [20]. These experiments illustrate that the tighter parameters developed here lead to better practical convergence rate.

The rest of the paper is organized as follows. Section 2 discusses related literature. We present the general class of problems that we address—bilinear saddle-point problems—and describe how they relate to EFGs in Section 3. Then Section 4 describes our optimization framework. Section 5 introduces treeplexes, the class of convex polytopes that define our domains of the optimization problems. Our focus is on dilated entropy-based DGFs; we introduce these in Section 6 and present our main results—bounds on the associated strong convexity parameter and treeplex diameter. In Section 7 we demonstrate the use of our results on instantiating EGT. We compare our approach with the current state-of-art in EFG solving and discuss the extent of theoretical improvements achievable via our approach in Section 7.1. Section 8 presents numerical experiments testing the effect of various parameters on the performance of our approach as well as comparing the performance of our approach to CFR and CFR+. We close with a summary of our results and a few compelling further research directions in Section 9.

2 Related Literature

Nash equilibrium computation has received extensive attention in the literature [7, 8, 9, 13, 18, 25, 26, 39]. The equilibrium-finding problems vary quite a bit based on their characteristics; here we restrict our attention to two-player zero-sum sequential games.

Koller et al. [17] present an LP which has size linear in the size of the game tree. This approach, coupled with lossless abstraction techniques, was used to solve Rhode-Island hold'em [9, 34], a game with 3.1 billion nodes (roughly size $5 \cdot 10^7$ after lossless abstraction). However, for very large games, the resulting LPs tend to not fit in the computer memory, and iterations of the simplex algorithm or interior-point methods tend to be too slow. Bošanskỳ et al. [2] extend the LP approach by considering a form of column-and-row-generation. This algorithm scales only for games where it can identify an equilibrium of small support, and thus suffers from the same performance issues as the general LP approach. The scalability issues of LP-based approaches thus require approximate solution techniques. These techniques fall into two categories: iterative ε-Nash equilibrium-finding algorithms and game abstraction techniques [33].

The most popular iterative Nash equilibrium algorithms are variants of the counterfactual-regret-minimization framework [23, 36, 39]. All these algorithms operate by defining a notion of regret local to each information

[1] Confirmed through author communication

[2] This variation uses the current iterate rather than the average iterate due to decreased memory usage. It has inferior practical iteration complexity.

set, called *counterfactual regret*. A simplex-based regret-minimizing algorithm is then instantiated independently at each information set and given the counterfactual regrets for minimizing. It is shown that the per-information-set regret bounds obtained by the simplex-based regret minimizers lead to an overall bound on the convergence rate. Initially, the most practically popular variants were instantiated with *regret matching* (CFR) [39], and sometimes used Monte-Carlo methods for estimating regrets [23]. Recently a new regret-minimization technique called *regret matching plus* (RM+) was shown to be practically superior when coupled with a more aggressive stepsizing strategy [3, 36]. The algorithm combining RM+ and aggressive stepsizing is referred to as CFR+. Despite their slow convergence rate of $O(\frac{1}{\varepsilon^2})$, these regret-based algorithms perform very well in practice, especially CFR+. Recently, Waugh and Bagnell [38] showed, with some caveats, an interpretation of CFR as a FOM with $O(\frac{1}{\varepsilon^2})$ rate. In particular, they show that CFR is equivalent to *dual averaging* [31] (when using a particular set of distance functions that fall in the general class of dilated distance functions [12]) on the simplex (and thereby also on information sets with no child information sets), whereas on internal information sets these algorithms differ in how they aggregate utilities from child information sets. It is still unknown whether CFR is fully equivalent to a first-order method. Despite these similarities, in this paper we make a distinction between regret-based methods and $O(\frac{1}{\varepsilon})$ FOMs for ease of exposition when comparing algorithmic methodologies.

Hoda et al. [12] introduce DGFs for EFGs leading to $O(\frac{1}{\varepsilon})$ convergence rate when used with EGT. Kroer et al. [20] improved the parameters associated with these dilated entropy function based DGFs. While Gilpin et al. [10] give an algorithm with convergence rate $O(\ln(\frac{1}{\varepsilon}))$, their bound has a dependence on a certain condition number of the payoff matrix, which is difficult to estimate, and their bound independent of the condition number has a $O(\frac{1}{\varepsilon})$ convergence rate. We compare all three algorithms discussed here in detail in Section 7.1. Based on our results, Kroer et al. [21] show how to extend FOMs and our DGF to the computation of approximate Nash-equilibrium refinements. Brown et al. [6] show experimentally that certain parts of the game tree can be pruned when computing gradients for EGT with our DGF.

3 Problem setup

3.1 Basic notation

We let $\langle x, y\rangle$ denote the standard inner product of vectors x, y. Given a vector $x \in \mathbb{R}^n$, we let $\|x\|_p$ denote its ℓ_p norm given by $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$ for $p \in [1,\infty)$ and $\|x\|_\infty := \max_{i\in[n]} |x_i|$ for $p = \infty$. Throughout this paper, we use Matlab notation to denote vector and matrices, i.e., $[x; y]$ denotes the concatenation of two column vectors x, y. Given $n \in \mathbb{N}$, we denote the simplex $\Delta_n := \{x \in \mathbb{R}^n_+ : \sum_{i=1}^n x_i = 1\}$. For a given set Q, we let $\mathrm{ri}\,(Q)$ denote its relative interior.

3.2 Sequence form

Computing a Nash equilibrium in a two-player zero-sum EFG with perfect recall can be formulated as a Bilinear Saddle Point Problem (BSPP):

$$\min_{x\in\mathscr{X}} \max_{y\in\mathscr{Y}} \langle x, Ay\rangle = \max_{y\in\mathscr{Y}} \min_{x\in\mathscr{X}} \langle x, Ay\rangle. \tag{1}$$

This is known as the *sequence-form* formulation [17, 32, 37] in the EFG literature. In this formulation, x and y correspond to the nonnegative strategy vectors for players 1 and 2 and the sets $\mathscr{X}, \mathscr{Y}$ are convex polyhedral reformulations of the sequential strategy space of these players. Here $\mathscr{X}, \mathscr{Y}$ are defined by the constraints $Ex = e, Fy = f$, where each row of E, F encodes part of the sequential nature of the strategy vectors, the right hand-side vectors e, f are $|\mathscr{I}_1|, |\mathscr{I}_2|$-dimensional vectors, and $\mathscr{I}_i$ is the information sets

for player i. The matrix A encodes the reward structure associated with the game. For a complete treatment of this formulation, see von Stengel [37].

Our theoretical developments mainly exploit the treeplex domain structure and are independent of other structural assumptions resulting from EFGs. Therefore, we describe our results for general BSPPs. We follow the presentation and notation of Juditsky and Nemirovski [14, 15] for BSPPs. For notation and presentation of treeplex structure, we follow Kroer et al. [20].

4 Optimization setup

In its most general form a BSPP is defined as

$$\mathrm{Opt} := \max_{y\in\mathscr{Y}} \min_{x\in\mathscr{X}} \phi(x,y), \tag{$\mathscr{S}$}$$

where $\mathscr{X},\mathscr{Y}$ are nonempty convex compact sets in Euclidean spaces $\mathbf{E}_x,\mathbf{E}_y$ and $\phi(x,y) = \upsilon + \langle a_1,x\rangle + \langle a_2,y\rangle + \langle y,Ax\rangle$. We let $\mathscr{Z} := \mathscr{X}\times\mathscr{Y}$; so $\phi(x,y) : \mathscr{Z}\to\mathbb{R}$. In the context of EFG solving, $\phi(x,y)$ is simply the inner product given in (1).

The BSPP $(\mathscr{S})$ gives rise to two convex optimization problems that are dual to each other:

$$\begin{array}{rcll} \mathrm{Opt}(P) & = & \min_{x\in\mathscr{X}}[\overline{\phi}(x) := \max_{y\in\mathscr{Y}}\phi(x,y)] & (P), \\ \mathrm{Opt}(D) & = & \max_{y\in\mathscr{Y}}[\underline{\phi}(y) := \min_{x\in\mathscr{X}}\phi(x,y)] & (D), \end{array}$$

with $\mathrm{Opt}(P) = \mathrm{Opt}(D) = \mathrm{Opt}$. It is well known that the solutions to $(\mathscr{S})$ — the saddle points of ϕ on $\mathscr{X}\times\mathscr{Y}$ — are exactly the pairs $z=[x;y]$ comprised of optimal solutions to the problems (P) and (D). We quantify the accuracy of a candidate solution $z=[x;y]$ with the *saddle point residual*

$$\varepsilon_{\mathrm{sad}}(z) \quad := \quad \overline{\phi}(x) - \underline{\phi}(y) = \underbrace{\left[\overline{\phi}(x) - \mathrm{Opt}(P)\right]}_{\geq 0} + \underbrace{\left[\mathrm{Opt}(D) - \underline{\phi}(y)\right]}_{\geq 0}.$$

In the context of EFG, $\varepsilon_{\mathrm{sad}}(z)$ measures the proximity to being an ε-Nash equilibrium.

4.1 General framework for FOMs

Most FOMs capable of solving BSPP $(\mathscr{S})$ are quite flexible in terms of adjusting to the geometry of the problem characterized by the domains $\mathscr{X},\mathscr{Y}$ of the BSPP $(\mathscr{S})$. The following components are standard in forming the setup for such FOMs (we present components for $\mathscr{X}$, analogous components are used for $\mathscr{Y}$):

- *Vector norm*: $\|\cdot\|_{\mathscr{X}}$ on the Euclidean space $\mathbf{E}_x$ where the domain $\mathscr{X}$ of $(\mathscr{S})$ lives, along with its dual norm $\|\zeta\|_{\mathscr{X}}^* = \max\limits_{\|x\|_{\mathscr{X}}\leq 1}\langle\zeta,x\rangle$.
- *Matrix norm*: $\|A\| = \max_y\left\{\|Ay\|_{\mathscr{X}}^* : \|y\|_{\mathscr{Y}} = 1\right\}$ based on the vector norms $\|\cdot\|_{\mathscr{X}}$ and $\|\cdot\|_{\mathscr{Y}}$.
- *Distance-Generating Function* (DGF): A function $\omega_{\mathscr{X}}(x) : \mathscr{X}\to\mathbb{R}$, which is convex and continuous on $\mathscr{X}$, and admits a continuous selection of subgradients $\omega'_{\mathscr{X}}(x)$ on the set $\mathscr{X}^\circ := \{x\in\mathscr{X} : \partial\omega_{\mathscr{X}}(x)\neq\emptyset\}$ (here $\partial\omega_{\mathscr{X}}(x)$ is a subdifferential of $\omega_{\mathscr{X}}$ taken at x), and is strongly convex with modulus $\varphi_{\mathscr{X}}$ w.r.t. the norm $\|\cdot\|_{\mathscr{X}}$:

$$\forall x',x''\in\mathscr{X}^\circ:\ \langle\omega'_{\mathscr{X}}(x') - \omega'_{\mathscr{X}}(x''),x'-x''\rangle \geq \varphi_{\mathscr{X}}\|x'-x''\|_{\mathscr{X}}^2. \tag{2}$$

- *Bregman distance*: $V(u\|x) := \omega_{\mathscr{X}}(u) - \omega_{\mathscr{X}}(x) - \langle\omega'_{\mathscr{X}}(x),u-x\rangle$ for all $x\in\mathscr{X}^\circ$ and $u\in\mathscr{X}$.

- *Prox-mapping*: Given a *prox center* $x \in \mathscr{X}^\circ$,

$$\underset{x}{\text{Prox}}(\xi) := \underset{u \in \mathscr{X}}{\text{argmin}} \{\langle \xi, u \rangle + V(u\|x)\} : \mathbf{E} \to \mathscr{X}^\circ.$$

  For properly chosen stepsizes, the prox-mapping becomes a contraction. This is critical in the convergence analysis of FOMs. Furthermore, when the DGF is taken as the squared ℓ_2 norm, the prox mapping becomes the usual projection operation of the vector $x - \xi$ onto $\mathscr{X}$.

- ω-*center*: $x_\omega := \underset{x \in \mathscr{X}}{\text{argmin}}\, \omega_{\mathscr{X}}(x) \in \mathscr{X}^\circ$ of $\mathscr{X}$.

- *Set width*: $\Omega_x := \max_{x \in \mathscr{X}} V(x\|x_\omega) \leq \max_{x \in \mathscr{X}} \omega_{\mathscr{X}}(x) - \min_{x \in \mathscr{X}} \omega_{\mathscr{X}}(x)$.

In this paper, for ease of exposition, we will introduce and work with the *Excessive Gap Technique* (EGT) of Nesterov [29] as the FOM. Next we introduce the related terminology and formally state the algorithm and the results from [29].

Nesterov [30] introduced *smoothed approximations* to the functions $\overline{\phi}$ and $\underline{\phi}$ via the distance-generating functions $\omega_{\mathscr{X}}, \omega_{\mathscr{Y}}$ as follows:

$$\overline{\phi}_{\mu_2}(x) = \max_{y \in \mathscr{Y}} \{\phi(x,y) - \mu_2 \omega_{\mathscr{Y}}(y)\}, \tag{3}$$

$$\underline{\phi}_{\mu_1}(y) = \min_{x \in \mathscr{X}} \{\phi(x,y) + \mu_1 \omega_{\mathscr{X}}(x)\}, \tag{4}$$

where $\mu_1, \mu_2 > 0$ are smoothness parameters denoting the amount of smoothing applied. Let $y_{\mu_2}(x)$ and $x_{\mu_1}(y)$ be the y and x solutions attaining the optima in (3) and (4). These can be thought of as *smoothed best responses*. Nesterov [30] also established that the gradients of the functions $\overline{\phi}_{\mu_2}(x)$ and $\underline{\phi}_{\mu_1}(y)$ exist and are Lipschitz continuous. The gradient operators and Lipschitz constants are given as follows

$$\nabla \overline{\phi}_{\mu_2}(x) = a_1 + A y_{\mu_2}(x) \quad \text{and} \quad \nabla \underline{\phi}_{\mu_1}(y) = a_2 + A^\top x_{\mu_1}(y),$$

$$L_1\left(\overline{\phi}_{\mu_2}\right) = \frac{\|A\|^2}{\varphi_{\mathscr{Y}} \mu_2} \quad \text{and} \quad L_2\left(\underline{\phi}_{\mu_1}\right) = \frac{\|A\|^2}{\varphi_{\mathscr{X}} \mu_1}.$$

Based on this setup, we formally state the EGT of [29] in Algorithm 1.

ALGORITHM 1: EGT

input : ω-centers x_ω, y_ω, smoothness weights μ_1, μ_2, and desired accuracy ε

output: $z^t = [x^t, y^t]$

$t = 0,\ \mu_1^t = \mu_1,\ \mu_2^t = \mu_2$;

$x^t = \text{Prox}_{x_\omega}\left(\mu_1^{-1} \nabla \overline{\phi}_{\mu_2}(x_\omega)\right)$;

$y^t = y_{\mu_2}(x_\omega)$;

while $\varepsilon_{\text{sad}}(z^t) > \varepsilon$ **do**

  $\tau_t = \frac{2}{t+3}$;

  if *t is even* **then**

    $(\mu_1^{t+1}, x^{t+1}, y^{t+1}) = Step(\mu_1^t, \mu_2^t, x^t, y^t, \tau_t)$

  else

    $(\mu_2^{t+1}, y^{t+1}, x^{t+1}) = Step(\mu_2^t, \mu_1^t, y^t, x^t, \tau_t)$

  $t = t + 1$;

ALGORITHM 2: Step

input : μ_1, μ_2, x, y, τ

output: μ_1^+, x_+, y_+

$\hat{x} = (1-\tau)x + \tau x_{\mu_1}(y)$;

$y_+ = (1-\tau)y + \tau y_{\mu_2}(\hat{x})$;

$\tilde{x} = \text{Prox}_{x_{\mu_1}(y)}\left(\frac{\tau}{(1-\tau)\mu_1} \nabla \overline{\phi}_{\mu_2}(\hat{x})\right)$;

$x_+ = (1-\tau)x + \tau \tilde{x}$;

$\mu_1^+ = (1-\tau)\mu_1$;

The EGT algorithm alternates between taking steps focused on $\mathscr{X}$ and $\mathscr{Y}$. Algorithm 2 shows a single step focused on $\mathscr{X}$. Steps focused on y are completely analogous. Algorithm 1 shows how initial points are selected and the alternating steps and stepsizes are computed. Nesterov [29] proves that the EGT algorithm converges in $O(\frac{1}{\varepsilon})$ steps to an ε approximate saddle point of the BSPP ($\mathscr{S}$). More precisely, Nesterov [29, Theorem 6.3] states the following:

Theorem 1. *Suppose the input values μ_1, μ_2 in the EGT algorithm satisfy*

$$\mu_1 = \frac{\varphi_{\mathscr{X}}}{L_1(\overline{\phi}_{\mu_2})} = \frac{\varphi_{\mathscr{X}}\varphi_{\mathscr{Y}}}{\|A\|^2}\mu_2.$$

Then, at every iteration $t \geq 1$ of the EGT algorithm, the corresponding solution $z^t = [x^t; y^t]$ satisfies $x^t \in \mathscr{X}$, $y^t \in \mathscr{Y}$, and

$$\overline{\phi}(x^t) - \underline{\phi}(y^t) = \varepsilon_{\text{sad}}(z^t) \leq \frac{4\|A\|}{t+1}\sqrt{\frac{\Omega_{\mathscr{X}}\Omega_{\mathscr{Y}}}{\varphi_{\mathscr{X}}\varphi_{\mathscr{Y}}}}.$$

5 Treeplexes

Hoda et al. [12] introduce the *treeplex*, a class of convex polytopes that encompass the sequence-form description of strategy spaces in perfect-recall EFGs. Understanding the treeplex structure is crucial because the proofs of our main results rely on induction over these structures.

Definition 1. *Treeplexes are defined recursively as follows:*

1. Basic sets*: The standard simplex Δ_m is a treeplex.*
2. Cartesian product*: If $Q_1, \ldots, Q_k$ are treeplexes, then $Q_1 \times \cdots \times Q_k$ is a treeplex.*
3. Branching*: Given a treeplex $P \subseteq [0,1]^p$, a collection of treeplexes $Q = \{Q_1, \ldots, Q_k\}$ where $Q_j \subseteq [0,1]^{n_j}$, and $l = \{l_1, \ldots, l_k\} \subseteq \{1, \ldots, p\}$, the set defined by*

$$P\boxed{l}Q := \left\{(u, q_1, \ldots, q_k) \in \mathbb{R}^{p+\sum_j n_j} : \; u \in P,\; q_1 \in u_{l_1} \cdot Q_1, \ldots, q_k \in u_{l_k} \cdot Q_k\right\}$$

is a treeplex. We say u_{l_j} is the branching variable for the treeplex Q_j.

A treeplex is a tree of simplexes where children are connected to their parents through the branching operation. In the branching operation, the child simplex domain is scaled by the value of the parent branching variable. For EFGs, the simplexes correspond to the information sets of a single player and the whole treeplex represents that player's strategy space. The branching operation has a sequential interpretation: The vector u represents the decision variables at certain stages, while the vectors q_j represent the decision variables at the k potential following stages, depending on external outcomes. Here $k \leq p$ since some variables in u may not have subsequent decisions. For treeplexes, von Stengel [37] has suggested a polyhedral representation of the form $Eu = e$ where the matrix E has its entries from $\{-1, 0, 1\}$ and the vector e has its entries in $\{0, 1\}$.

For a treeplex Q, we denote by S_Q the index set of the set of simplexes contained in Q (in an EFG S_Q is the set of information sets belonging to the player). For each $j \in S_Q$, the treeplex rooted at the j-th simplex Δ^j is referred to as Q_j. Given vector $q \in Q$ and simplex Δ^j, we let $\mathbb{I}_j$ denote the set of indices of q that correspond to the variables in Δ^j and define q^j to be the sub vector of q corresponding to the variables in $\mathbb{I}_j$. For each simplex Δ^j and branch $i \in \mathbb{I}_j$, the set $\mathscr{D}_j^i$ represents the set of indices of simplexes reached immediately after Δ^j by taking branch i (in an EFG $\mathscr{D}_j^i$ is the set of potential next-step information sets for

the player). Given a vector $q \in Q$, simplex Δ^j, and index $i \in \mathbb{I}_j$, each child simplex Δ^k for every $k \in \mathscr{D}_j^i$ is scaled by q_i. Conversely, for a given simplex Δ^j, we let p_j denote the index in q of the parent branching variable q_{p_j} that Δ^j is scaled by. We use the convention that $q_{p_j} = 1$ if Q is such that no branching operation precedes Δ^j. For each $j \in S_Q$, d_j is the depth of the treeplex rooted at Δ^j, that is, the maximum number of edges between Δ^j and any simplex beneath Δ^j plus one (that is, a lone simplex has depth 1, not 0). In an EFG the depth is the length of the longest sequence of actions starting at Δ^j. Then d_Q gives the depth of Q. We use b_Q^j to identify the number of branching operations preceding the j-th simplex in Q. We will say that a simplex j such that $b_Q^j = 0$ is a *root simplex*.

Figure 1 illustrates an example treeplex Q. This treeplex Q is constructed from nine two-to-three-dimensional simplexes $\Delta^1, \ldots, \Delta^9$. At level 1, we have two root simplexes, Δ^1, Δ^2, obtained by a Cartesian product operation (denoted by $\times$). We have maximum depths $d_1 = 2$, $d_2 = 1$ beneath them. Since there are no preceding branching operations, the parent variables for these simplexes Δ^1 and Δ^2 are $q_{p_1} = q_{p_2} = 1$. For Δ^1, the corresponding set of indices in the vector q is $\mathbb{I}_1 = \{1,2\}$, while for Δ^2 we have $\mathbb{I}_2 = \{3,4,5\}$. At level 2, we have the simplexes $\Delta^3, \ldots, \Delta^7$. The parent variable of Δ^3 is $q_{p_3} = q_1$; therefore, Δ^3 is scaled by the parent variable q_{p_3}. Similarly, each of the simplexes $\Delta^3, \ldots, \Delta^7$ is scaled by their parent variables q_{p_j} that the branching operation was performed on. So on for Δ^8 and Δ^9 as well. The number of branching operations required to reach simplexes Δ^1, Δ^3 and Δ^8 is $b_Q^1 = 0, b_Q^3 = 1$ and $b_Q^8 = 2$, respectively.

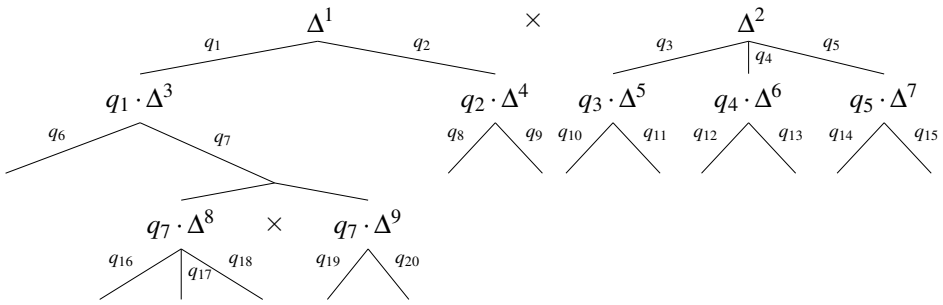

Figure 1: An example treeplex constructed from 9 simplexes. Cartesian product operation is denoted by $\times$.

Note that we allow more than two-way branches; hence our formulation follows that of Kroer et al. [20] and differs from that of Hoda et al. [12]. As discussed in Hoda et al. [12], it is possible to model sequence-form games by treeplexes that use only two-way branches. Yet, this can cause a large increase in the depth of the treeplex, thus leading to significant degradation in the associated strong convexity parameter. Because we handle multi-way branches directly in our framework, our approach is more effective in taking into account the structure of the sequence-form game and thereby resulting in better bounds on the associated strong convexity parameters and thus overall convergence rates.

Our analysis requires a measure of the size of a treeplex Q. For this purpose, we define $M_Q \coloneqq \max_{q \in Q} \|q\|_1$.

In the context of EFGs, suppose Q encodes player 1's strategy space; then M_Q is the maximum number of information sets with nonzero probability of being reached when player 1 has to follow a pure strategy while the other player may follow a mixed strategy. We also let

$$M_{Q,r} \coloneqq \max_{q \in Q} \sum_{j \in S_Q : b_Q^j \leq r} \|q^j\|_1. \tag{5}$$

Intuitively, $M_{Q,r}$ gives the maximum value of the ℓ_1 norm of any vector $q \in Q$ after removing the variables corresponding to simplexes that are not within r branching operations of the root of Q.

The quantities M_Q and $|S_Q|$ play an important role in the comparison of our bounds. We discuss them on an example next.

Example 1. *In order to illustrate M_Q and compare it to the size of $|S_Q|$, let us now consider an example of an EFG and its corresponding treeplexes. Consider a game where two players take turns choosing among k actions, and each player chooses actions d times before leaf nodes are reached. In the treeplex Q of Player* 1*, each time Player* 1 *chooses among k actions constitutes a size k branching operation, and every time Player* 2 *chooses among k actions constitutes a size k Cartesian product operation. The total dimensionality of the treeplex, $|S_Q|$, is k^{2d}, while the value of M_Q is k^d (since only Cartesian products blow up). Thus, M_Q is square root of the size of $|S_Q|$.*

6 Dilated entropy functions with bounded strong convexity

In this section we introduce DGFs for domains with treeplex structures and establish their strong convexity parameters with respect to a given norm (see (2)).

The basic building block in our construction is the *entropy* DGF given by $\omega_e(z) = \sum_{i=1}^n z_i \log(z_i)$, for the simplex Δ_n. It is well-known that $\omega_e(\cdot)$ is strongly convex with modulus 1 with respect to the ℓ_1 norm on Δ_n (see Juditsky and Nemirovski [14]). We will show that a suitable modification of this function achieves a desirable strong convexity parameter for the treeplex domain.

The treeplex structure is naturally related to the *dilation operation* [11] defined as follows: Given a compact set $K \subseteq \mathbb{R}^d$ and a function $f : K \to \mathbb{R}$, we first define

$$\bar{K} := \left\{ (t,z) \in \mathbb{R}^{d+1} : \; t \in [0,1] \,,\; z \in t \cdot K \right\}.$$

Definition 2. *Given a function $f(z)$, the* dilation operation *is defined as the function $\bar{f} : \bar{K} \to \mathbb{R}$ given by*

$$\bar{f}(z,t) = \begin{cases} t \cdot f(z/t) & \text{if } t > 0 \\ 0 & \text{if } t = 0 \end{cases}.$$

The dilation operation preserves convexity, and thus the following function defined based on dilating the entropy function over the simplexes of a treeplex is convex:

Definition 3. *Given a treeplex Q and weights $\beta_j > 0$ for each $j \in S_Q$, we define the* dilated entropy function *as*

$$\omega(q) = \sum_{j \in S_Q} \beta_j \sum_{i \in \mathbb{I}_j} q_i \log \frac{q_i}{q_{p_j}} \quad \text{for any } q \in Q,$$

where we follow the treeplex notation and p_j is the index of the branching variable preceding Δ^j, with the convention that $q_{p_j} = 1$ if Δ^j has no branching operation preceding it.

Remark 1. *Note that the dilated entropy function $\omega(\cdot)$ defined above is twice differentiable in the relative interior of treeplex Q and admits a continuous gradient selection. Moreover, for weights β_j that scale appropriately with depth d_j, we will demonstrate that it is strongly convex w.r.t. the ℓ_1 norm. Thus, the dilated entropy function is compatible with the ℓ_1 norm, as required by the BSPP setup.* ∎

We would also like the prox-mapping associated with our DGF to be efficiently computable. Hoda et al. [12] show that for any dilated function, its prox operator on a treeplex can be easily computed through a recursive bottom-up traversal involving the prox mappings associated with the function being dilated on

individual simplexes. Since the entropy prox function can be computed in closed form on a simplex, the dilated entropy function can be computed by a single treeplex traversal involving closed-form expressions on each simplex.

Definition 3 above leads to a subset of the DGFs considered by Hoda et al. [12]. Hoda et al. [12] introduce the general class of treeplex DGFs obtained by dilating any simplex DGF, and show that any such DGF is guaranteed to have some lower bound on the strong-convexity parameter. They show explicit bounds for a special class of treeplexes called *uniform treeplexes*. Kroer et al. [20] showed that stronger bounds can be obtained for the dilated entropy DGF on general treeplexes through careful selection of the weights. Our main theoretical result shows that by selecting the weights β_j according to a particular recurrence, we can significantly improve the strong convexity bounds associated with the dilated entropy function, and we show that our analysis is tight.

We will consider weights that satisfy the following recurrence:

$$\begin{aligned} &\alpha_j = 1 + \max_{i \in \mathbb{I}_j} \sum_{k \in \mathscr{D}_j^i} \frac{\alpha_k \beta_k}{\beta_k - \alpha_k}, && \forall j \in S_Q, \\ &\beta_j > \alpha_j, && \forall j \in S_Q \text{ s.t. } b_Q^j > 0, \\ &\beta_j = \alpha_j, && \forall j \in S_Q \text{ s.t. } b_Q^j = 0. \end{aligned} \tag{6}$$

Intuitively, α_j represents the negative terms that the weight β_j has to cancel out: the constant 1 represents the negative term resulting from the squared norm in the strong convexity requirement; the summation term represents the amount of negative terms accumulated from the induction on simplexes descending from simplex j. The qualifications on β_j ensure that β_j is set such that it at least cancels out the negative terms; the difference $\beta_j - \alpha_j$ controls the amount of negative value the parent simplex has to make up. When $b_Q^j = 0$ there is no parent simplex, and so we set $\beta_j = \alpha_j$. The reason for our requirement of a strict inequality $\beta_j > \alpha_j$ for non-root simplexes becomes evident in the proof of Lemma 2.

Based on recurrence (6), our main results establish strong convexity of our dilated entropy DGF w.r.t. the ℓ_2 and ℓ_1 norms:

Theorem 2. *For a treeplex Q, the dilated entropy function with weights satisfying recurrence (6) is strongly convex with modulus* 1 *with respect to the ℓ_2 norm.*

Theorem 3. *For a treeplex Q, the dilated entropy function with weights satisfying recurrence (6) is strongly convex with modulus $\frac{1}{M_Q}$ with respect to the ℓ_1 norm.*

We give the proofs of Theorems 2 and 3 in Section 6.2. Based on Theorem 3, we get the following corollary:

Corollary 1. *For a treeplex Q, the dilated entropy function with weights $\beta_j = 2 + \sum_{r=1}^{d_j} 2^r (M_{Q_j,r} - 1)$ for all $j \in S_Q$ is strongly convex with modulus $\frac{1}{M_Q}$ w.r.t. the ℓ_1 norm.*

Corollary 1 follows easily from Theorem 3 and a recursive interpretation of the weights, which is presented as Fact 2 in the next section. In particular, a specific choice of weights in Fact 2 immediately satisfies the recurrence (6) and leads to Corollary 1.

To our knowledge, the best strong convexity bounds for general treeplexes were proved in Kroer et al. [20]. Using weights $\beta_j = 2^{d_j} M_{Q_j}$ they show strong convexity modulus $\frac{1}{|S_Q|}$ w.r.t. the ℓ_1 norm; see [20, Theorem 5.4]. Corollary 1 improves the prior bounds by exchanging a factor of $|S_Q|$ with a factor of M_Q. Note that $|S_Q|$ is tied to the branching factor associated with branching operations in the treeplex Q whereas

M_Q is not. Thus, our result removes the dependence of the strong convexity parameter on the branching factor and hence significantly improves upon Kroer et al. [20].

In Theorem 4 we use our strong convexity result to establish a polytope diameter that has only a logarithmic dependence on the branching factor. As a consequence, the associated dilated entropy DGF when used in FOMs such as MP and EGT for solving EFGs leads to the same improvement in their convergence rate.

6.1 Preliminary results for the proofs of our main results

We start with some simple facts and a few technical lemmas that are used in our proofs.

Fact 1. *Given a treeplex Q, we have, respectively, for all $i \in \mathbb{I}_j, j \in S_Q$ and all $d = 1, \ldots, d_Q, q \in Q$:*

$$(a) \quad M_{Q_j} \geq 1 + \sum_{l \in \mathscr{D}_j^i} M_{Q_l}, \qquad\qquad (b) \quad M_Q \geq \sum_{j \in S_Q : d_j = d} q_{p_j} M_{Q_j}.$$

Proof. The first inequality was established in Kroer et al. [20, Lemma 5.7]. The second follows by using $M_Q = \sum_j q_i$ for some q, and inductively replacing terms belonging to simplexes j at the bottom with M_{Q_j}. The result follows because branching operations cancel out by summing to 1. □

□

Our next observation follows from Fact 1(a) and is advantageous in suggesting a practically useful choice of the weights β_j that can be used for Theorem 3 to arrive at Corollary 1.

Fact 2. *Let Q be a treeplex and $\beta_j = 2 + \sum_{r=1}^{d_j} 2^r (M_{Q_j,r} - 1)$ for all $j \in S_Q$ as in Corollary 1. Then Fact 1(a) implies $\beta_j \geq 2 + \sum_{k \in \mathscr{D}_j^i} 2\beta_k, \ \ \forall i \in \mathbb{I}_j$ and $\forall j \in S_Q$.*

Consequently, by selecting $\beta_j = 2\alpha_j$, and $\alpha_j = 1 + \sum_{r=1}^{d_j} 2^{r-1}(M_{Q_j,r} - 1)$ for all $i \in \mathbb{I}_j$ and for all $j \in S_Q$ such that $b_Q^j > 0$, we immediately satisfy the conditions of the recurrence in (6).

Given a twice differentiable function f, we let $\nabla^2 f(z)$ denote its Hessian at z. Our analysis is based on the following sufficient condition for strong convexity of a twice differentiable function:

Fact 3. *A twice-differentiable function f is strongly convex with modulus φ with respect to a norm $\|\cdot\|$ on nonempty convex set $C \subset \mathbb{R}^n$ if $h^\top \nabla^2 f(z) h \geq \varphi \|h\|^2, \ \forall h \in \mathbb{R}^n, z \in C^\circ$.*

For simplexes Δ^j at depth 1, there is no preceding branching operation; so the variables h_{p_j}, q_{p_j} do not exist. We circumvent this with the convention $h_{p_j} = 0, q_{p_j} = 1$ for such $j \in S_Q$.

In our proofs we will use the expression derived in Lemma 1 for $h^\top \nabla^2 \omega(q) h$.

Lemma 1. *Given a treeplex Q and a dilated entropy function $\omega(\cdot)$ with weights $\beta_j > 0$, we have*

$$h^\top \nabla^2 \omega(q) h = \sum_{j \in S_Q} \beta_j \left[\sum_{i \in \mathbb{I}_j} \left(\frac{h_i^2}{q_i} - \frac{2 h_i h_{p_j}}{q_{p_j}} \right) + \frac{h_{p_j}^2}{q_{p_j}} \right] \quad \forall q \in \mathrm{ri}\,(Q) \text{ and } \forall h \in \mathbb{R}^n. \tag{7}$$

Proof. Consider $q \in \mathrm{ri}\,(Q)$ and any $h \in \mathbb{R}^n$. For each $j \in S_Q$ and $i \in \mathbb{I}_j$, the second-order partial derivates of $\omega(\cdot)$ w.r.t. q_i are:

$$\nabla^2_{q_i^2} \omega(q) = \frac{\beta_j}{q_i} + \sum_{k \in \mathscr{D}_j^i} \sum_{l \in \mathbb{I}_k} \frac{\beta_k q_l}{q_i^2} = \frac{\beta_j}{q_i} + \sum_{k \in \mathscr{D}_j^i} \frac{\beta_k}{q_i}, \tag{8}$$

where the last equality holds because $k \in \mathscr{D}_j^i$ and thus $\sum_{l\in\mathbb{I}_k} q_l = \|q^k\|_1 = q_{p_k} = q_i$. Also, for each $j \in S_Q, i \in \mathbb{I}_j, k \in \mathscr{D}_j^i$, and $l \in \mathbb{I}_k$, the second-order partial derivates w.r.t. q_i, q_l are given by:

$$\nabla^2_{q_i,q_l}\omega(q) = \nabla^2_{q_l,q_i}\omega(q) = -\frac{\beta_k}{q_i}. \tag{9}$$

Then equations (8) and (9) together imply

$$h^\top \nabla^2 \omega(q) h = \sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\left[h_i^2\left(\frac{\beta_j}{q_i} + \sum_{k\in\mathscr{D}_j^i}\frac{\beta_k}{q_i}\right) - \sum_{k\in\mathscr{D}_j^i}\sum_{l\in\mathbb{I}_k} h_i h_l \frac{2\beta_k}{q_i}\right]. \tag{10}$$

Given $j \in S_Q$ and $i \in \mathbb{I}_j$, we have $p_k = i$ for each $k \in \mathscr{D}_j^i$ and for any $k \in \mathscr{D}_j^i$, there exists some other $j' \in S_Q$ corresponding to k in the outermost summation. Then we can rearrange the following terms:

$$\sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j} h_i^2 \sum_{k\in\mathscr{D}_j^i}\frac{\beta_k}{q_i} = \sum_{j\in S_Q}\beta_j\frac{h_{p_j}^2}{q_{p_j}} \quad\text{and}\quad \sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\sum_{k\in\mathscr{D}_j^i}\sum_{l\in\mathbb{I}_k} h_i h_l\frac{2\beta_k}{q_i} = \sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\beta_j\frac{2h_i h_{p_j}}{q_{p_j}}.$$

Using these two equalities in the relation (10) leads to (7) and proves the lemma. □

□

6.2 Proofs of our main theorems

The majority of the work for our strong-convexity results is performed by the following lemma, from which our strong convexity results follow easily.

Lemma 2. *For any treeplex Q, the dilated entropy function with weights satisfying recurrence* (6) *satisfies the following inequality:*

$$h^\top \nabla^2 \omega(q) h \geq \sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i} \quad \forall q \in \mathrm{ri}\,(Q) \text{ and } \forall h \in \mathbb{R}^n. \tag{11}$$

Proof. We will prove this by induction. First we will prove an inductive hypothesis over the set of non-root simplexes $\widehat{S}_Q = \left\{ j \in S_Q \;\middle|\; b_Q^j > 0 \right\}$. In order to state our inductive hypothesis we will need a notion of the set of simplexes currently at the "top" of the recursion: for a given depth d, we let the set of simplexes at the top be $\widehat{S}_Q^d = \left\{ k \in \widehat{S}_Q \;\middle|\; d_k \leq d, \exists j, i \text{ s.t. } k \in \mathscr{D}_j^i, d_j > d \right\}$. This is simply the set of simplexes such that their depth is less than d and the depth of their parent simplex is strictly greater than d. The reader may wonder why we do not perform the induction over simplices such that $d_j = d$. This is in order to avoid some technicalities relating to cases where two child simplexes from the same parent have different depths. By using $\widehat{S}_Q^d$, we ensure that the right-hand side of the inductive hypothesis always consists of the simplexes that are at the top of the treeplex given the current induction depth. We now show the following inductive hypothesis for any depth $d \geq 1$:

$$\sum_{j\in\widehat{S}_Q : d_j \leq d}\beta_j\left[\sum_{i\in\mathbb{I}_j}\left(\frac{h_i^2}{q_i} - \frac{2h_i h_{p_j}}{q_{p_j}}\right) + \frac{h_{p_j}^2}{q_{p_j}}\right] - \sum_{j\in\widehat{S}_Q : d_j \leq d}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i} \geq -\sum_{j\in\widehat{S}_Q^d}\frac{\beta_j\alpha_j}{\beta_j - \alpha_j}\frac{h_{p_j}^2}{q_{p_j}}.$$

We first show the inductive step, as the base case will follow from the same logic. Consider a treeplex Q of depth $d > 1$. By applying the inductive hypothesis we have

$$\sum_{j\in\widehat{S}_Q:d_j\le d}\beta_j\left[\sum_{i\in\mathbb{I}_j}\left(\frac{h_i^2}{q_i}-\frac{2h_ih_{p_j}}{q_{p_j}}\right)+\frac{h_{p_j}^2}{q_{p_j}}\right]-\sum_{j\in\widehat{S}_Q:d_j\le d}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i}$$
$$\ge\sum_{j\in\widehat{S}_Q:d_j=d}\beta_j\left[\sum_{i\in\mathbb{I}_j}\left(\frac{h_i^2}{q_i}-\frac{2h_ih_{p_j}}{q_{p_j}}\right)+\frac{h_{p_j}^2}{q_{p_j}}\right]-\sum_{j\in\widehat{S}_Q:d_j=d}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i}-\sum_{j\in\widehat{S}_Q^{d-1}}\frac{\beta_j\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}}. \tag{12}$$

Now we can rearrange terms: First we split $\widehat{S}_Q^{d-1}$ into two sets $\widehat{S}_Q^{d-1}\cap\widehat{S}_Q^d$ and $\widehat{S}_Q^{d-1}\setminus\widehat{S}_Q^d$. The sum over $j\in\widehat{S}_Q^{d-1}\setminus\widehat{S}_Q^d$ is equivalent to a sum over the immediate descendant information sets $k\in\mathscr{D}_j^i$ inside the square brackets since for each such $k\in\widehat{S}_Q^{d-1}\setminus\widehat{S}_Q^d$ there exists some $j\in\widehat{S}_Q^d$ such that $d_j=d$ (otherwise k would be in $\widehat{S}_Q^d$). Ignoring $\widehat{S}_Q^{d-1}\cap\widehat{S}_Q^d$ in (12) for now, we can write

$$\sum_{j\in\widehat{S}_Q:d_j=d}\beta_j\left[\sum_{i\in\mathbb{I}_j}\left(\frac{h_i^2}{q_i}-\frac{2h_ih_{p_j}}{q_{p_j}}\right)+\frac{h_{p_j}^2}{q_{p_j}}\right]-\sum_{j\in\widehat{S}_Q:d_j=d}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i}-\sum_{j\in\widehat{S}_Q^{d-1}\setminus\widehat{S}_Q^d}\frac{\beta_j\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}}. \tag{13}$$
$$=\sum_{j\in\widehat{S}_Q:d_j=d}\beta_j\left[\sum_{i\in\mathbb{I}_j}\left(\frac{h_i^2}{q_i}-\frac{2h_ih_{p_j}}{q_{p_j}}-\sum_{k\in\mathscr{D}_j^i}\frac{\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}}\right)+\frac{h_{p_j}^2}{q_{p_j}}\right]-\sum_{j\in\widehat{S}_Q:d_j=d}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i}.$$

Now we split the term $\frac{h_{p_j}^2}{q_{p_j}}$ into separate terms multiplied by $\frac{q_i}{q_{p_j}}$ and move it inside the parentheses by using the fact that $\sum_{i\in\mathbb{I}_j}\frac{q_i}{q_{p_j}}=1$, this gives

$$=\sum_{j\in\widehat{S}_Q:d_j=d}\beta_j\left[\sum_{i\in\mathbb{I}_j}\left(\frac{h_i^2}{q_i}-\frac{2h_ih_{p_j}}{q_{p_j}}-\sum_{k\in\mathscr{D}_j^i}\frac{\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}}+\frac{q_ih_{p_j}^2}{q_{p_j}^2}\right)\right]-\sum_{j\in\widehat{S}_Q:d_j=d}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i}.$$

Now we can move the sum over $i\in\mathbb{I}_j$ outside the square brackets and consolidate the summation terms to get

$$=\sum_{j\in\widehat{S}_Q:d_j=d}\sum_{i\in\mathbb{I}_j}\left[\left(\beta_j-1-\sum_{k\in\mathscr{D}_j^i}\frac{\beta_k\alpha_k}{\beta_k-\alpha_k}\right)\frac{h_i^2}{q_i}-\left(\frac{2\beta_jh_ih_{p_j}}{q_{p_j}}\right)+\frac{q_i\beta_jh_{p_j}^2}{q_{p_j}^2}\right]$$
$$\ge\sum_{j\in\widehat{S}_Q:d_j=d}\sum_{i\in\mathbb{I}_j}\left[(\beta_j-\alpha_j)\frac{h_i^2}{q_i}-\left(\frac{2\beta_jh_ih_{p_j}}{q_{p_j}}\right)+\frac{q_i\beta_jh_{p_j}^2}{q_{p_j}^2}\right], \tag{14}$$

where the last inequality follows from the definition of α_j.

For indices $j\in S_Q$ such that $b_Q^j>0$ and $i\in\mathbb{I}_j$, the relations in (6) imply $\beta_j>\alpha_j$, and so the expression inside the square brackets in (14) is a convex function of h_i. Taking its derivative w.r.t. h_i and setting it to zero gives $h_i=\frac{\beta_j}{\beta_j-\alpha_j}\frac{q_i}{q_{p_j}}h_{p_j}$. Thus, we arrive at

$$(14)\ge\sum_{j\in\widehat{S}_Q:d_j=d}\sum_{i\in\mathbb{I}_j}\left[\frac{\beta_j^2}{\beta_j-\alpha_j}\frac{q_ih_{p_j}^2}{q_{p_j}^2}-\frac{\beta_j^2}{\beta_j-\alpha_j}\frac{2q_ih_{p_j}^2}{q_{p_j}^2}+\frac{q_i\beta_jh_{p_j}^2}{q_{p_j}^2}\right]$$
$$=\sum_{j\in\widehat{S}_Q:d_j=d}\frac{h_{p_j}^2}{q_{p_j}}\left[\left(\frac{-\beta_j^2}{\beta_j-\alpha_j}+\beta_j\right)\frac{\sum_{i\in\mathbb{I}_j}q_i}{q_{p_j}}\right]=-\sum_{j\in\widehat{S}_Q:d_j=d}\frac{\beta_j\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}}. \tag{15}$$

Now we take our lower bound (15) on (13) and apply it to (12). Noting that $\widehat{S}_Q^d = \left\{\widehat{S}_Q : d_j = d\right\} \cup \left\{\widehat{S}_Q^{d-1} \cap \widehat{S}_Q^d\right\}$ we get

$$(12) \geq -\sum_{j\in\widehat{S}_Q : d_j = d} \frac{\beta_j\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}} - \sum_{j\in\widehat{S}_Q^{d-1}\cap\widehat{S}_Q^d} \frac{\beta_j\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}} = -\sum_{j\in\widehat{S}_Q^d} \frac{\beta_j\alpha_j}{\beta_j-\alpha_j}\frac{h_{p_j}^2}{q_{p_j}} \tag{16}$$

Hence, the induction step is complete. For the base case $d = 0$ we do not need the inductive assumption: Because $\mathscr{D}_j^i = \emptyset$, $\alpha_j = 1$, and we get (14) by definition; we can then apply the same convexity argument. This proves our inductive hypothesis.

Then using Lemma 1, we now have

$$\begin{aligned} h^\top\nabla^2\omega(q)h - \sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i} &= \sum_{j\in S_Q}\beta_j\left[\sum_{i\in\mathbb{I}_j}\left(\frac{h_i^2}{q_i} - \frac{2h_ih_{p_j}}{q_{p_j}}\right) + \frac{h_{p_j}^2}{q_{p_j}}\right] - \sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i} \\ &\geq \sum_{j\in S_Q : b_Q^j = 0}\left[\sum_{i\in\mathbb{I}_j}\beta_j\frac{h_i^2}{q_i} - \sum_{k\in\mathscr{D}_j^i}\frac{\beta_k\alpha_k}{\beta_k-\alpha_k}\frac{h_i^2}{q_i} - \frac{h_i^2}{q_i}\right] \geq 0. \end{aligned}$$

The first inequality follows from the fact that $h_{p_j} = 0$ for all $j \in S_Q$ such that $b_Q^j = 0$, and for all $j \in S_Q$ such that $b_Q^j > 0$, we used our induction. The last inequality follows from (6) and $q_i, h_i^2 \geq 0$. This then proves (11). □

□

We are now ready to prove our two main theorems, which we restate before proving them.

Theorem 2. *For a treeplex Q, the dilated entropy function with weights satisfying recurrence (6) is strongly convex with modulus* 1 *with respect to the ℓ_2 norm.*

Proof. Since $q_i \leq 1$, Lemma 2 implies $h^\top\nabla^2\omega(q)h \geq \sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j} h_i^2 = \|h\|_2^2$ for all $q \in \operatorname{ri}(Q)$ and for all $h \in \mathbb{R}^n$. Because the dilated entropy function $\omega(q)$ is twice differentiable on $\operatorname{ri}(Q)$, from Fact 3, we conclude that $\omega(\cdot)$ is strongly convex w.r.t. the ℓ_2 norm on Q with modulus 1. □

□

Remark 2. *Note that the analysis in the proof of Theorem 2 is tight. By choosing a vector $q \in \{0,1\}^{|Q|}$ such that $\|q\|_1 = M_Q$, and setting $h_i = \frac{\beta_j}{\beta_j-\alpha_j}\frac{q_i}{q_{p_j}}h_{p_j}$ for all indices i such that $q_i = 1$ and $h_i = 0$ otherwise, every inequality in the proof of Lemma 2 becomes an equality.* ∎

Theorem 3. *For a treeplex Q, the dilated entropy function with weights satisfying recurrence (6) is strongly convex with modulus $\frac{1}{M_Q}$ with respect to the ℓ_1 norm.*

Proof. To show strong convexity with modulus 1 w.r.t. the ℓ_1 norm, we lower bound the right-hand side of (11) in Lemma 2:

$$\sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i} \geq \frac{1}{M_Q}\left(\sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j} q_i\right)\sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\frac{h_i^2}{q_i} \geq \frac{1}{M_Q}\left(\sum_{j\in S_Q}\sum_{i\in\mathbb{I}_j}\frac{|h_i|}{\sqrt{q_i}}\sqrt{q_i}\right)^2 = \frac{1}{M_Q}\|h\|_1^2,$$

where the first inequality follows from the fact that M_Q is an upper bound on $\|q\|_1$ for any $q \in Q$, and the second inequality follows from the Cauchy-Schwarz inequality.

Hence, we deduce $h^\top \nabla^2 \omega(q) h \geq \frac{1}{M_Q}\|h\|_1^2$ holds for all $q \in \mathrm{ri}\,(Q)$ and for all $h \in \mathbb{R}^n$. Because the dilated entropy function $\omega(q)$ is twice differentiable on $\mathrm{ri}\,(Q)$, from Fact 3, we conclude that $\omega(\cdot)$ is strongly convex w.r.t. the ℓ_1 norm on Q with modulus $\varphi = \frac{1}{M_Q}$. □

□

6.3 Treeplex width

The convergence rates of FOMs such as MP and EGT algorithms depend on the diameter-to-strong convexity parameter ratio $\frac{\Omega}{\varphi}$, as described in Section 4.1. In order to establish full results on the convergence rates of these FOMs, we now bound this ratio using Corollary 1 scaled by M_Q.

Theorem 4. *For a treeplex Q, the dilated entropy function with simplex weights $\beta_j = M_Q(2+\sum_{r=1}^{d_j} 2^r(M_{Q_j,r}-1))$ for each $j \in S_Q$ results in $\frac{\Omega}{\varphi} \leq M_Q^2 2^{d_Q+2}\log m$ where m is the dimension of the largest simplex Δ^j for $j \in S_Q$ in the treeplex structure.*

Proof. For our choice of scaled weights β_j, Corollary 1 implies that the resulting dilated entropy function is strongly convex with modulus $\varphi = 1$. Hence, we only need to bound Ω.

Any $q \in Q$ satisfying $q_i \in \{0,1\}$ for all i maximizes $\omega(q)$ and results in $\max_{q\in Q}\omega(q) = 0$. For the minimum value, consider any $q \in \mathrm{ri}\,(Q)$. Applying the well-known lower bound of $-\log m$ for the negative entropy function on an m-dimensional simplex gives

$$\begin{aligned}
\omega(q) &= \sum_{j\in S_Q} \beta_j q_{p_j} \sum_{i\in\mathbb{I}_j} \frac{q_i}{q_{p_j}} \log \frac{q_i}{q_{p_j}} \geq -\sum_{j\in S_Q} \beta_j q_{p_j} \log m = -\sum_{d=0}^{d_Q} \sum_{j\in S_Q : d_j = d} \beta_j q_{p_j} \log m \\
&= -\sum_{d=1}^{d_Q} \sum_{j\in S_Q : d_j = d} \beta_j q_{p_j} \log m - \sum_{j\in S_Q : d_j = 0} \beta_j q_{p_j} \log m \\
&= -M_Q \log m \sum_{d=1}^{d_Q} \sum_{j\in S_Q : d_j = d} q_{p_j} \left(2 + \sum_{r=1}^{d} 2^r (M_{Q_j,r} - 1)\right) - M_Q \sum_{j\in S_Q : d_j = 0} 2 q_{p_j} \log m \\
&\geq -M_Q \log m \sum_{d=1}^{d_Q} \sum_{j\in S_Q : d_j = d} q_{p_j} M_{Q_j} \sum_{r=1}^{d} 2^r - 2M_Q \log m \sum_{j\in S_Q : d_j = 0} q_{p_j}, \qquad (17)
\end{aligned}$$

where the last inequality follows because for each $j \in S_Q$ with $d_j = 0$, the definition of M_Q implies $\sum_{j\in S_Q : d_j = 0} q_{p_j} \leq M_Q$, and for each $j \in S_Q$ with $d_j = d \geq 1$, we have $2 + \sum_{r=1}^{d} 2^r (M_{Q_j,r} - 1) \leq \sum_{r=1}^{d} 2^r M_{Q_j,r} \leq \sum_{r=1}^{d} 2^r M_{Q_j}$ since $M_{Q_j,r} \leq M_{Q_j}$. Also, from Fact 1(b), we have $\sum_{j\in S_Q : d_j = d} q_{p_j} M_{Q_j} \leq M_Q$. Then we arrive at

$$\begin{aligned}
(17) &\geq -M_Q^2 \log m \left(2 + \sum_{d=1}^{d_Q} \sum_{r=1}^{d} 2^r\right) = -M_Q^2 \log m \left(2 + \sum_{d=1}^{d_Q} (2^{d+1} - 2)\right) \\
&= -M_Q^2 \log m \left(2 + \sum_{d=1}^{d_Q} 2^{d+1} - 2d_Q\right) \geq -M_Q^2 (\log m) 2^{d_Q+2},
\end{aligned}$$

where the last inequality follows because for $d_Q = 0$ we have $2^{d_Q+2} = 4 > 2$ and for $d_Q \geq 1$ we have $2d_Q \geq 2$.

This lower bound on the minimum value, i.e., $\min_{q\in Q}\omega(q) \geq -M_Q^2(\log m)2^{d_Q+2}$, coupled with $\max_{q\in Q}\omega(q) \leq 0$, establishes the theorem. □

□

7 EGT for extensive-form game solving

We now describe how to instantiate EGT for solving two-player zero-sum EFGs of the form (1) with treeplex domains. Below we state the customization of all the definitions from Section 4 for our problem.

Let m be the size of the largest simplex in either of the treeplexes $\mathscr{X}, \mathscr{Y}$. Because $\mathscr{X}$ and $\mathscr{Y}$ are treeplexes, they are closed, convex, and bounded. We use the ℓ_1 norm on both of the embedding spaces $\mathbf{E}_x, \mathbf{E}_y$. As our DGFs for $\mathscr{X}, \mathscr{Y}$ compatible with the ℓ_1 norm, we use the dilated entropy DGF scaled with weights given in Theorem 4. Then Theorem 4 gives our bound on $\frac{\Omega_{\mathscr{X}}}{\varphi_{\mathscr{X}}}$ and $\frac{\Omega_{\mathscr{Y}}}{\varphi_{\mathscr{Y}}}$. Because the dual norm of the ℓ_1 norm is the ℓ_∞ norm, the matrix norm is given by $\|A\| = \max_{y \in \mathscr{Y}} \{\|Ay\|_1^* : \|y\|_1 = 1\} = \max_{i,j} |A_{i,j}|$.

Remark 3. *Note that $\|A\|$ is not at the scale of the maximum payoff difference in the original game. The values in A are scaled by the probability of the observed nature outcomes on the path of each sequence. Thus, $\|A\|$ is exponentially smaller (in the number of observed nature steps on the path to the maximizing sequence) than the maximum payoff difference in the original EFG.* ∎

Theorem 4 immediately leads to the following convergence rate result for FOMs equipped with dilated entropy DGFs to solve EFGs (and more generally BSPPs over treeplex domains).

Theorem 5. *Consider a BSPP over treeplex domains $\mathscr{X}, \mathscr{Y}$. Then EGT algorithm equipped with the dilated entropy DGF with weights $\beta_j = 2 + \sum_{r=1}^{d_j} 2^r (M_{\mathscr{X}_j, r} - 1)$ for all $j \in S_{\mathscr{X}}$ and the corresponding setup for $\mathscr{Y}$ will return an ε-accurate solution to the BSPP in at most the following number of iterations:*

$$\frac{\max_{i,j} |A_{i,j}| \sqrt{M_{\mathscr{X}}^2 2^{d_{\mathscr{X}}+2} M_{\mathscr{Y}}^2 2^{d_{\mathscr{Y}}+2}} \log m}{\varepsilon}.$$

This rate in Theorem 5, to our knowledge, establishes the state-of-the-art for FOMs with $O(\frac{1}{\varepsilon})$ convergence rate for EFGs.

7.1 Improvements in extensive-form game convergence rate

The ratio $\frac{\Omega}{\varphi}$ of set diameter over the strong convexity parameter is important for FOMs that rely on a prox function, such as EGT and MP. Compared to the rate obtained by [20], we get the following improvement: for simplicity, assume that the number of actions available at each information set is on average a, then our bound improves the convergence rate of [20] by a factor of $\Omega(d_{\mathscr{X}} \cdot a^{d_{\mathscr{X}}} + d_{\mathscr{Y}} \cdot a^{d_{\mathscr{Y}}})$.

As mentioned previously, Hoda et al. [12] proved only explicit bounds for the special case of uniform treeplexes that are constructed as follows: 1) A base treeplex Q_b along with a subset of b indices from it for branching operations is chosen. 2) At each depth d, a Cartesian product operation of size k is applied. 3) Each element in a Cartesian product is an instance of the base treeplex with a size b branching operation leading to depth $d-1$ uniform treeplexes constructed in the same way. Given bounds Ω_b, φ_b for the base treeplex, the bound of Hoda et al. [12] for a uniform treeplex with d uniform treeplex levels (note that the total depth of the constructed treeplex is $d \cdot d_{Q_b}$, where d_{Q_b} is the depth of the base treeplex Q_b) is

$$\frac{\Omega}{\varphi} \leq O\left(b^{2d-2} k^{2d+2} d^2 M_{Q_b}^2 \frac{\Omega_b}{\varphi_b}\right).$$

Then when the base treeplex is a simplex of dimension m, their bound for the dilated entropy on a uniform treeplex Q becomes

$$\frac{\Omega}{\varphi} \leq O\left(|S_Q|^2 d_Q^2 \log m\right).$$

Even for the special case of a uniform treeplex with a base simplex, comparing Theorem 4 to their bound, we see that our general bound improves the associated constants by exchanging $O(|S_Q|^2 d_Q^2)$ with $O(M_Q^2 2^{d_Q})$. Since M_Q does not depend on the branching operation in the treeplex, whereas $|S_Q|$ does, our bounds are also the first bounds to remove an exponential dependence on the branching operation (we have only a logarithmic dependence). In Example 1 we showed that there exist games where $M_Q = \sqrt{|S_Q|}$, and in general M_Q is much smaller than $|S_Q|$. Consequently, our results establish the best known convergence results for all FOMs based on dilated entropy DGF such as EGT, MP, and stochastic variants of FOMs for BSPPs.

Gilpin et al. [10] give an equilibrium-finding algorithm presented as $O(\ln(\frac{1}{\varepsilon}))$; but this form of their bound has a dependence on a certain condition number of the A matrix. Specifically, their iteration bound for sequential games is $O(\frac{\|A\|_{2,2}\cdot\ln(\|A\|_{2,2}/\varepsilon)\cdot\sqrt{D}}{\delta(A)})$, where $\delta(A)$ is the condition number of A, $\|A\|_{2,2} = \sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$ is the Euclidean matrix norm, and $D = \max_{x,\bar{x}\in\mathscr{X},y,\bar{y}\in\mathscr{Y}} \|[x;y]-[\bar{x};\bar{y}]\|_2^2$. Unfortunately, the condition number $\delta(A)$ is only shown to be finite for these games. Without any such unknown quantities based on condition numbers, Gilpin et al. [10] establish a convergence rate of $O(\frac{\|A\|_{2,2}\cdot D}{\varepsilon})$. This algorithm, despite having the same dependence on ε as ours in its convergence rate, i.e., $O(\frac{1}{\varepsilon})$, suffers from worse constants. In particular, there exist matrices such that $\|A\|_{2,2} = \sqrt{\|A\|_{1,\infty}\|A\|_{\infty,1}}$, where $\|A\|_{1,\infty}$ and $\|A\|_{\infty,1}$ correspond to the maximum absolute column and row sums, respectively. Then together with the value of D, this leads to a cubic dependence on the dimension of Q. For games where the players have roughly equal-size strategy spaces, this is equivalent to a constant of $O(M_Q^4)$ as opposed to our constant of $O(M_Q^2)$. In addition, as compared with previous work, the authors also only show experiments with their algorithm on instances with $9\cdot 10^6$ leaf nodes [10], whereas the previous EGT algorithm showed experiments on instances with up to $4\cdot 10^{12}$ leaf nodes.

CFR, CFR+, and EGT all need to keep track of a constant number of current and/or average iterates, so the memory usage of all three algorithms is of the same order. When gradients are computed using tree traversal as opposed to storing the matrix A (or some decomposition thereof), each of these algorithms require a constant times the number of sequences in the sequence-form representation; in particular, each algorithm needs to keep track of some current $x \in \mathscr{X}$ and $y \in \mathscr{Y}$ iterate, as well as a small number of gradients and intermediate solutions. Therefore, we compare mainly the number of iterations required by each algorithm. Since the theoretical properties of CFR and CFR+ are comparable, we compare to CFR, with all statements being valid for CFR+ as well.

CFR has a $O(\frac{1}{\varepsilon^2})$ convergence rate; but its dependence on the number of information sets is only linear (and sometimes sublinear [23]). Since our results utilizing EGT have a quadratic dependence on M_Q^2, CFR sometimes has a better dependence on game constants and can be more attractive for obtaining low-quality solutions quickly for games with many information sets. However, our theoretical results could be coupled with a $O(\frac{1}{\varepsilon^2})$ convergence rate FOM such as mirror descent in order to achieve a similar dependence on game constants. In practice, a sampling-based variant of CFR called Monte-Carlo CFR (MCCFR) is preferred for certain applications [23]. MCCFR and CFR have a similar convergence rate, though MCCFR has cheaper iterations. Using the gradient-sampling method described by Kroer et al. [20], our theoretical results can be utilized with the stochastic mirror prox algorithm [16] in order to achieve the same iteration cost and convergence rate as MCCFR.

8 Numerical experiments

We carried out numerical experiments to investigate the practical performance of EGT on EFGs when instantiated with our DGF. We start out by comparing our DGF (henceforth referred to as new DGF) with that of Kroer et al. [20] (henceforth referred to as old DGF) when used in the EGT algorithm, and then

we compare EGT equipped with our DGF to CFR and CFR+, the practical state-of-the-art EFG-solving algorithms.

We consider two variants of EGT: the original algorithm of Nesterov [29] as is, and a new variant where we incorporate several heuristics for speeding up practical convergence by avoiding overly pessimistic parameters. We demonstrate and discuss the practical convergence issues in comparison experiments later. First we describe our practical variant.

Our preliminary experiments for EGT demonstrated that the initial values for the smoothing parameters μ_1, μ_2 are much too conservative in practice. Instead, in our aggressive EGT we follow an automated tuning procedure. The goal of this procedure is to find a pair of initial smoothing parameters μ_1, μ_2 that fit the problem instance at hand. At the beginning of the algorithm we perform a binary search over 16 logarithmically-spaced numbers between $\frac{0.001}{\Omega_x}$ and 1.0 for μ_1. For each number, we binary search 3 multiples of μ_1 in order to choose μ_2: $0.75, 1$, and 1.25. We choose the smallest pair of parameters that give an excessive gap greater than 0.001 after computing x^0, y^0 in Algorithm 1.

Because we are choosing the smoothing parameters this way, we are no longer guaranteed convergence according to the EGT theory; however we can fix this problem by numerically checking the excessive gap condition at every iteration. We perform this check as a part of the following more aggressive step-sizing policy: instead of setting $\tau = \frac{2}{t+3}$ at every iteration t, we use the *aggressive μ reduction* heuristic of Hoda et al. [12], where a constant stepsize is used, and then whenever the post-step check of the excessive gap condition fails the step is retraced and τ is halved. We start τ at $\frac{1}{2}$. In addition to only decreasing τ when the excessive gap check fails, we also increase it by a factor of 1.11 whenever a step was successful. We also introduce a new heuristic for checking whether a step was successful: we check whether the saddle-point residual $\varepsilon_{\text{sad}}(z)$ deteriorated by a factor of more than 1.1 after each step. If it did, we retrace and halve τ. This heuristic can be computed essentially for free (in particular using the gradients from the excessive gap check) and thus only requires two treeplex traversals.

We test the algorithms on two games. The first game is a scaled up variant of the poker game Leduc holdem [35], a benchmark problem in the imperfect-information game-solving community. In our version, the deck consists of k pairs of cards $1 \ldots k$, for a total deck size of $2k$. Each player initially pays one chip to the pot, and is dealt a single private card. After a round of betting, a community card is dealt face up. After a subsequent round of betting, if neither player has folded, both players reveal their private cards. If either player pairs their card with the community card they win the pot. Otherwise, the player with the highest private card wins. In the event both players have the same private card, they draw and split the pot. We consider decks with 6, 30, and 70 cards. The smallest game has about 2000 nodes in the game tree, and the largest has about 3.2 million nodes in the game tree.

The second game is a zero-sum variant of a search-game played on the graph shown in Figure 2, we will refer to it as *Search*. Search is a simultaneous-move game (which can be modeled as a turn-taking EFG with appropriately chosen information sets). A defender controls two patrols that can each move within their respective shaded areas (labeled P1 and P2), and at each time step the controller chooses a move for both patrols. An attacker tries to move from the S node to one of the three payoff nodes. The attacker can move freely to any adjacent node (except at patrolled nodes, the attacker cannot move from a patrolled node to another patrolled node). The attacker can also choose to wait in place for a time step in order to clean up their traces. If a patrol visits a node that was previously visited by the attacker, and the attacker did not wait to clean up their traces, they can see that the attacker was there. If the attacker reaches any of the rightmost nodes they received the respective payoff at the node (5, 10, or 3, respectively) and the defender loses that amount. If the attacker and any patrol are on the same node at any time step, the attacker is captured, which leads to payoffs of 1 and -1 for the defender and attacker respectively. Finally, the game times out after 5 simultaneous moves, in which case both players receive a payoff 0. Search has 87,927 nodes and 11,830 and 69 defender and attacker sequences. General-sum variants of this game were studied by Bošanskỳ and Čermák [1], Bošanskỳ et al. [2], and Kroer et al. [22]. One particularly noteworthy feature of this game

is that the strategy spaces are extremely imbalanced: it is huge for one player and tiny for the other. This is in stark contrast with Leduc (and other poker games) where the strategy spaces are almost the same. In addition to our general results that we are about to present, we believe that our results are the first of their kind in demonstrating strong FOM performance on such imbalanced EFGs.

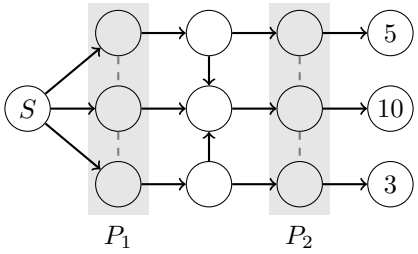

Figure 2: A search game between a defender and an attacker.

In our comparisons, we use loglog plots that show the results for a particular game. In each plot, we show the performance of the algorithms, with the x-axis showing the number of tree traversals, and the y-axis showing the solution accuracy, i.e. the saddle-point residual. We note that tree-traversals is a good proxy for overall computational effort because the majority of the time in the algorithms EGT and CFR algorithms for EFG solving is spent on gradient computations, which in our case directly translates into tree-traversals. All EGT experiments include the tree traversals needed for automated parameter tuning described in the previous paragraph as part of the number of tree traversals performed by the algorithm.

First, we investigate the impact of applying the weights used in recurrence (6), as compared to the previous scheme introduced in Kroer et al. [20]. To instantiate recurrence (6) we have to choose a way to set β_j relative to α_j. We use the scheme of Corollary 1. This scheme will henceforth be referred to as new weights. We compare these new weights to the weights used in Kroer et al. [20] (henceforth referred to as old weights). Figure 3 shows the result of running EGT with the old and the new weights for Leduc with a 6-card (on the left) deck and Search (on the right). The top row shows results when instantiating EGT with the parameters dictated by the theory, whereas the bottom row shows EGT using all our heuristics. When instantiating parameters according to theory the most important observation is that the parameters are way too pessimistic, out of thousands of gradient computations, the majority are spent decreasing the smoothing parameters until they are small enough that the algorithms start to make progress. That said, this happens significantly faster for the new parameters than the old ones; for the search game $20,000$ gradient computations is not enough to start progressing with the old parameters. For our aggressive EGT variant we find that both DGFs perform much better, though our new weights still perform better on both games, significantly so for Search.

We next compare the performance of EGT with our new weights to that of the CFR and CFR+ algorithms on three Leduc variants (6, 30, and 70-card decks) and Search. For CFR we use two variants: the vanilla CFR algorithm with RM as the regret minimizer, and CFR with the RM+ regret minimizer and alternating minimization. Finally we have CFR+ which adds linear stepsizing on top of RM+ and alternating minimization. The results are shown in Figure 4. We find that EGT instantiated with our DGF outperforms CFR with both RM and RM+ and alternating minimization, whereas CFR+ is slightly faster still. EGT maintains a stronger convergence rate across all iterations. It starts out slightly worse because its first iterate is shifted outward on the x-axis due to paying the upfront cost of our automated tuning based initialization. However, its convergence rate is immediately superior to CFR, and almost immediately overtakes both CFR algorithms.

The performance we get from EGT (with aggressive stepsizing) relative to CFR is in sharp contrast to the previous conventional practical wisdom in the field. In Kroer et al. [20] it was found that, while EGT has

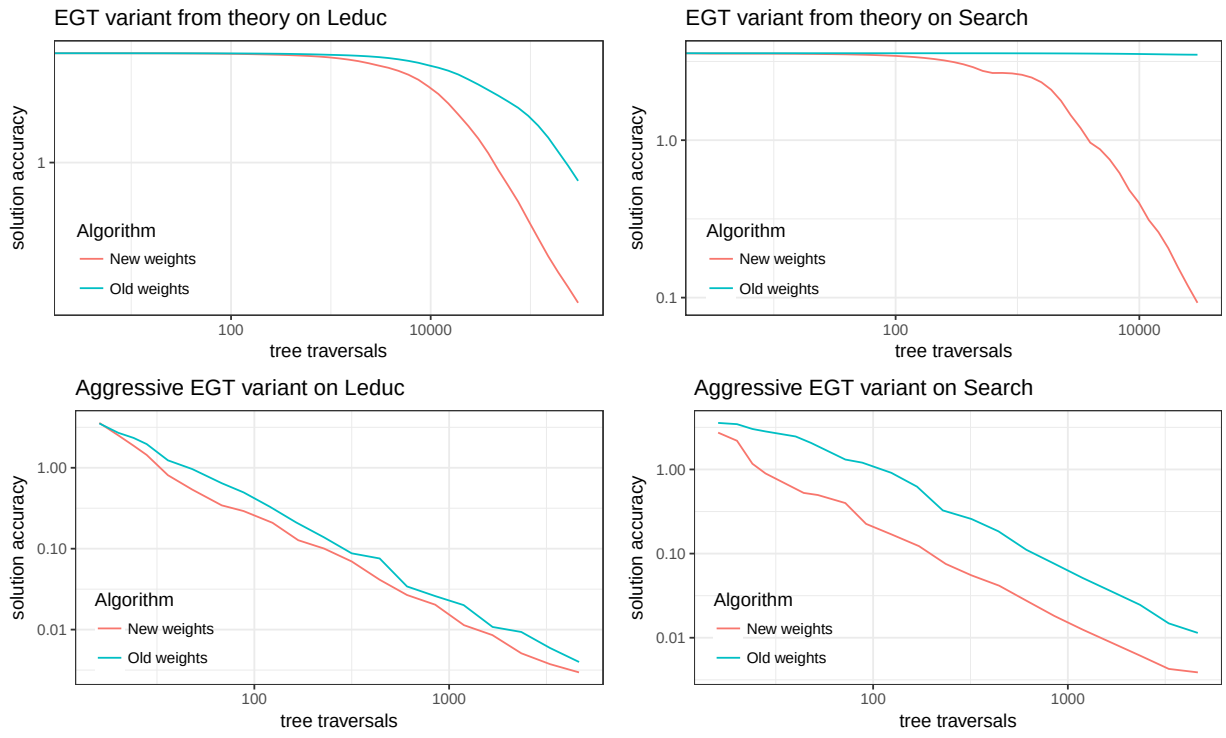

Figure 3: Solution accuracy as a function of the number of iterations for EGT with our weighting scheme (New weights) and with the weighting scheme from Kroer et al. [20] (Old weights). Both axes are on a log scale. The top row shows the effect of our weighting scheme when using EGT instantiated according to the original theory. The bottom row shows the effect when using our aggressive EGT variant.

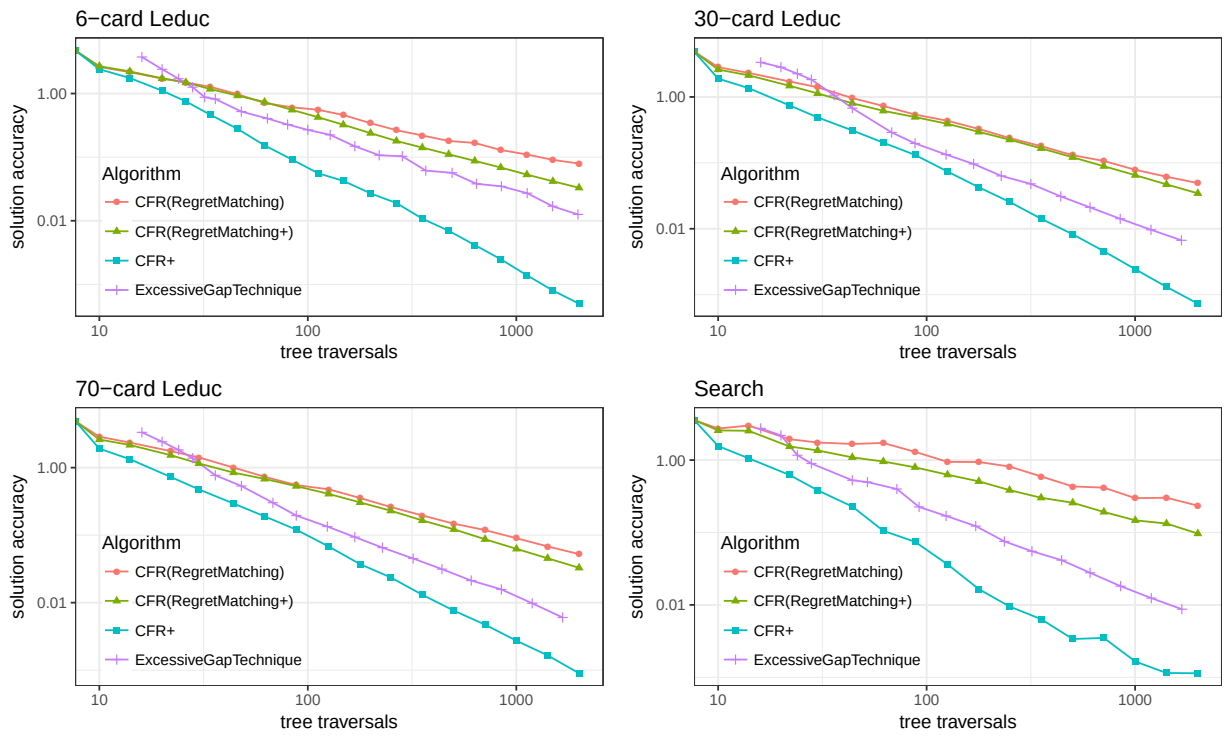

Figure 4: Solution accuracy as a function of the number of tree traversals in three different variants of Leduc hold'em and the Search game. Results are shown for CFR with regret mathing, CFR with regret mathing$^+$, CFR+, and our aggressive EGT algorithm. Both axes are shown on a log scale.

better asymptotic convergence rate, CFR had better initial convergence rate, and it was only after a certain number of iterations that EGT took over. Furthermore, the switching point where EGT is preferable was found to shift outward on the x-axis as the Leduc game size was increased. This sentiment has been mirrored by Brown and Sandholm [4]. In contrast to this, we find that our DGF along with proper initialization leads to EGT having a better convergence rate than not only CFR, but also CFR with RM+. Furthermore, scaling up the game size does not seem to adversely affect this relationship.

The numerical results in Figure 4 suggest that our DGF may be useful in practice for solving large-scale zero-sum EFGs. Moreover, these results are with a particular FOM, EGT, and there is a myriad of possible ways that our DGF could be combined with other FOMs.

9 Conclusions

We have investigated FOMs for computing Nash equilibria in two-player zero-sum perfect-recall EFGs. On the theoretical side, we analyzed the strong convexity properties of the dilated entropy DGF over treeplexes. By introducing specific weights that are tied to the structure of the treeplex, we improved prior results on treeplex diameter from $O(|S_Q|M_Q d 2^d \log m)$ to $O(M_Q^2 2^{d_Q+2} \log m)$, thereby removing all but a logarithmic dependence on branching associated with the branching operator in the treeplex definition. These results lead to significant improvements in the theoretical convergence rates of FOMs that can be equipped with dilated entropy DGFs and used for EFG solving including, but not limited to, EGT, MP, and Stochastic MP.

We numerically investigated the performance of EGT and compared it to the practical state-of-the-art algorithms CFR and CFR+. Our experiments showed that EGT equipped with the dilated entropy DGF, when tuned with a proper scaling, has better practical, as well as theoretical, convergence rate than CFR even with RM+, as opposed to the cross-over point phenomena based on the size of the games. While CFR+ is still faster, our results are for a specific FOM instantiated with our DGF; it seems likely that future experimental work could lead to even faster algorithms based on our DGF, for example by incorporating randomization to reduce the cost of gradient computation, or other FOMs.

Theorems 2 and 3 establish bounds for a general class of weights β_j satisfying the recurrence (6). Then in Corollary 1, we have selected a particular weighting scheme for β_j satisfying (6) and performed our numerical tests. There may be other interesting choices of β_j satisfying the recurrence (6). Thus, finding a way to optimally choose among the set of weights satisfying (6) to minimize the polytope diameter for specific games is appealing.

On a separate note, in practice CFR is often paired with an abstraction technique [33]. This is sometimes despite the lack of any theoretical justification. Effective ways to pair FOMs such as MP and EGT with practical abstraction techniques [5] or abstraction techniques that achieve solution-quality guarantees [18, 19, 24] are also worth further consideration.

References

[1] Branislav Bošanskỳ and Jiří Čermák. Sequence-form algorithm for computing stackelberg equilibria in extensive-form games. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[2] Branislav Bošanskỳ, Christopher Kiekintveld, Viliam Lisý, and Michal Pěchouček. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research*, 51:829–866, 2014.

[3] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, January 2015.

[4] Noam Brown and Tuomas Sandholm. Strategy-based warm starting for regret minimization in games. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 432–438, 2016.

[5] Noam Brown, Sam Ganzfried, and Tuomas Sandholm. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit texas hold'em agent. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 7–15. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[6] Noam Brown, Christian Kroer, and Tuomas Sandholm. Dynamic thresholding and pruning for regret minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 421–429, 2017.

[7] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1), 2009.

[8] Constantinos Daskalakis, Alan Deckelbaum, and Anthony Kim. Near-optimal no-regret algorithms for zero-sum games. *Games and Economic Behavior*, 92:327–348, 2015.

[9] Andrew Gilpin and Tuomas Sandholm. Lossless abstraction of imperfect information games. *Journal of the ACM*, 54(5), 2007.

[10] Andrew Gilpin, Javier Peña, and Tuomas Sandholm. First-order algorithm with $\mathcal{O}(\ln(1/\varepsilon))$ convergence for ε-equilibrium in two-person zero-sum games. *Mathematical Programming*, 133(1–2): 279–298, 2012. Conference version appeared in AAAI-08.

[11] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2001.

[12] Samid Hoda, Andrew Gilpin, Javier Peña, and Tuomas Sandholm. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2):494–512, 2010. Conference version appeared in WINE-07.

[13] Albert Jiang and Kevin Leyton-Brown. Polynomial-time computation of exact correlated equilibrium in compact games. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 119–126, 2011.

[14] Anatoli Juditsky and Arkadi Nemirovski. First order methods for nonsmooth convex large-scale optimization, i: general purpose methods. *Optimization for Machine Learning*, pages 121–148, 2011.

[15] Anatoli Juditsky and Arkadi Nemirovski. First order methods for nonsmooth convex large-scale optimization, ii: utilizing problems structure. *Optimization for Machine Learning*, pages 149–183, 2011.

[16] Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

[17] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259, 1996.

[18] Christian Kroer and Tuomas Sandholm. Extensive-form game abstraction with bounds. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, pages 621–638. ACM, 2014.

[19] Christian Kroer and Tuomas Sandholm. Imperfect-recall abstractions with bounds in games. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, pages 459–476. ACM, 2016.

[20] Christian Kroer, Kevin Waugh, Fatma Kılınç-Karzan, and Tuomas Sandholm. Faster first-order methods for extensive-form game solving. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, pages 817–834. ACM, 2015.

[21] Christian Kroer, Gabriele Farina, and Tuomas Sandholm. Smoothing method for approximate extensive-form perfect equilibrium. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[22] Christian Kroer, Gabriele Farina, and Tuomas Sandholm. Robust stackelberg equilibria in extensive-form games and extension to limited lookahead. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[23] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1078–1086, 2009.

[24] Marc Lanctot, Richard Gibson, Neil Burch, Martin Zinkevich, and Michael Bowling. No-regret learning in extensive-form games with imperfect recall. In *International Conference on Machine Learning (ICML)*, pages 65–72, 2012.

[25] Richard Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 36–41, San Diego, CA, 2003. ACM.

[26] Michael Littman and Peter Stone. A polynomial-time Nash equilibrium algorithm for repeated games. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 48–54, San Diego, CA, 2003.

[27] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in no-limit poker. *arXiv preprint arXiv:1701.01724*, 2017.

[28] Arkadi Nemirovski. Prox-method with rate of convergence $\mathcal{O}(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[29] Yurii Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal of Optimization*, 16(1):235–249, 2005.

[30] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103: 127–152, 2005.

[31] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[32] I. Romanovskii. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3:678–681, 1962.

[33] Tuomas Sandholm. The state of solving large incomplete-information games, and application to poker. *AI Magazine*, pages 13–32, Winter 2010. Special issue on Algorithmic Game Theory.

[34] Jiefu Shi and Michael Littman. Abstraction methods for game theoretic poker. In *CG '00: Revised Papers from the Second International Conference on Computers and Games*, pages 333–345, London, UK, 2002. Springer-Verlag.

[35] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 550–558, July 2005.

[36] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit texas hold'em. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 645–652, 2015.

[37] Bernhard von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

[38] Kevin Waugh and Drew Bagnell. A unified view of large-scale zero-sum equilibrium computation. In *Computer Poker and Imperfect Information Workshop at the AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[39] Martin Zinkevich, Michael Johanson, Michael H Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1729–1736, 2007.

A Notation and results described from an EFG perspective

In the body of the paper we described our results using notation oriented toward the convex-optimization perspective on treeplexes: the results are for general treeplexes, and so we view them as a general convex set. In this section we give an overview of our results from a standard EFG-specific perspective, in the hopes that it may be helpful for researchers who are familiar with that literature, but not the first-order methods literature and convex analysis.

First we define notation, then we give a description of how our treeplex results map onto traditional EFG notation, and finally we give a description of the dilated entropy smoothed best response tree traversal and EGT in terms of this notation.

A.1 Extensive-form games and the sequence-form

Extensive-form games (EFGs) can be thought of as a game tree, where each node in the tree corresponds to some history of actions taken by all players. Each node belongs to some player, and the actions available to the player at a given node are represented by the branches. Uncertainty is modeled by having a special player, *Nature*, that moves with some predefined fixed probability distribution over actions at each node belonging to Nature. EFGs model imperfect information by having groups of nodes in information sets, which is a group of nodes all belonging to the same player such that the player cannot distinguish among them. Finally we assume *perfect recall*, which requires that no player ever forgets their past actions (equivalently, for each information set there is only a single possible last action taken by the player to whom the information set belongs).

Definition 4. *A* two-player extensive-form game with imperfect information and perfect recall Γ *is a tuple* $(H, Z, A, P, \sigma_c, \mathscr{I}, u)$ *composed of:*

- H*: a finite set of possible sequences (or histories) of actions, such that the empty sequence* $\emptyset \in H$*, and every prefix* z *of* h *in* H *is also in* H*.*
- $Z \subseteq H$*: the set of terminal histories, i.e. those sequences that are not a proper prefix of any sequence.*
- A*: a function mapping* $h \in H \setminus Z$ *to the set of available actions at non-terminal history* h*.*

- *P: the player function, mapping each non-terminal history $h \in H \setminus Z$ to $\{1, 2, c\}$, representing the player who takes action after h. If $P(h) = c$, the player is Nature.*
- *σ_0: a function assigning to each $h \in H \setminus Z$ such that $P(h) = 0$ a probability mass function over $A(h)$.*
- *$\mathscr{I}_i$, for $i \in \{1, 1\}$: partition of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ for each h, h' in the same set of the partition. For notational convenience, we will write $A(I)$ to mean $A(h)$ for any of the $h \in I$, where $I \in \mathscr{I}_i$. $\mathscr{I}_i$ is the information partition of player i, while the sets in $\mathscr{I}_i$ are called the information sets of player i.*
- *u_i: utility function mapping $z \in Z$ to the utility (a real number) gained by player i when the terminal history is reached.*

We further assume that all players have perfect recall.

A strategy for a player i is usually represented in *behavioral form*, which consists of probability distributions over actions at each information set in $\mathscr{I}_i$. In this paper we will focus on an alternative, but strategically equivalent, representation of the set of strategies, called the *sequence form* [17, 32, 37]. In the sequence form, actions are instead represented by *sequences*. A sequence σ_i, is an ordered list of actions taken by player i on the path to some history h. In perfect-recall games, all nodes in an information set $I \in \mathscr{I}_i$ correspond to the same sequence for player i, we let $\text{seq}(I)$ denote this sequence. Given a sequence σ_i and an action a that Player i can take immediately after σ_i, we let $\sigma_i a$ denote the resulting new sequence. Instead of directly choosing the probability to put on an action, in the sequence form the probability of playing the entire sequence is chosen, this is called the *realization probability* and is denoted by $r_(\sigma_i)$. A choice of realization probabilities for every sequence belonging to Player i is called a *realization plan* and is denoted $r_i : \Sigma_i \to [0, 1]^{|\Sigma_i|}$. This representation relies on perfect recall: for any information set $I \in \mathscr{I}_i$ we have that each action $a \in A(I)$ is uniquely represented by a single sequence $\sigma_i = \text{seq}(I)a$, since $\text{seq}(I)$ corresponds to exactly one sequence. In particular, this gives us a simple way to convert any strategy in sequence form to a behavioral strategy: the probability of playing action $a \in A(I)$ at information set I is simply $\frac{r_i(\text{seq}(I)a)}{r_i(\text{seq}(I))}$.

A.2 Mapping between convex analysis notation and EFG notation

First we describe the treeplex. A treeplex Q is used to model the sequence-form strategy space of each player. Thus in an EFG, the treeplexes would be the sets of realization probabilities Σ_1, Σ_2. In the BSPP (1) the typical representation would be that $\mathscr{X} = \Sigma_1$ and $\mathscr{Y} = \Sigma_2$. For the remainder we will describe the notation in terms of Player 1 and Σ_1. The set of simplexes in Σ_1, with indices denoted by S_{Σ_1}, is the set of information sets $\mathscr{I}_1$ where Player 1 acts. The set $\mathscr{D}_j^i$ of simplexes reached immediately after taking branch i in simplex j is the set of potential information sets where Player 1 may have to act next. Which one is reached of course depends on which actions are taken by Nature and Player 2. Another way to put this is that $\mathscr{D}_j^i$ corresponds to the set of information sets $I \in \mathscr{I}_1$ such that $\sigma_1(I) = \sigma_1$, where σ_1 is the sequence corresponding to taking the action i at simplex j. The table below gives an overview of treeplex notation and how it corresponds to EFG notation and concepts. Not all our concepts are easily mapped to existing EFG ideas. For example M_Q and $M_{Q,r}$, the maximum ℓ_1 norm of Q and the r-depth-limited maximum ℓ_1 norm, are still most easily thought of in terms of norms. It is the maximum number of information sets with nonzero probability of being reached when player 1 has to follow a pure strategy while the other player may follow a mixed strategy. Intuitively the maximum ℓ_1 norm of Σ_1 measures the branching factor associated with observable opponent actions and Nature actions that cause Player 1 to reach different information sets, while *not* measuring branching factor associated with Player 1 choosing actions at an information set (since the ℓ_1 norm sums to one at such information set).

Treeplex notation	EFG meaning
Q	Σ_1, the set of realization probabilities
S_{Σ_1}	$\mathscr{I}_1$, the set of information-set indices into the treeplex Σ_1
$\mathscr{D}_j^i$	Set of information sets in Σ_1 such that the sequence corresponding to branch i at simplex j is the parent sequence
q_{p_j}	The parent sequence $\sigma_1(I_j)$, where I_j is the information set corresponding to simplex j
d_j	The length of the longest possible sequence of actions starting at I_j
$b_{\Sigma_1}^j$	The length of $\sigma_1(I_j)$

Our dilated entropy construction using the weights described in recurrence (6) can now be described in terms of EFG notation as follows:

$$\begin{aligned} &\alpha_j = 1 + \max_{a \in A_{I_j}} \sum_{k \in \mathscr{D}_{I_j}^a} \frac{\alpha_k \beta_k}{\beta_k - \alpha_k}, && \forall I_j \in \mathscr{I}_1, \\ &\beta_j > \alpha_j, && \forall I_j \in \mathscr{I}_1 \text{ s.t. } length(\sigma_1(I_j)) > 0, \\ &\beta_j = \alpha_j, && \forall I_j \in \mathscr{I}_1 \text{ s.t. } length(\sigma_1(I_j)) = 0. \end{aligned} \tag{18}$$

If we then instantiate EGT with this DGF and use Theorem 3 we get the following convergence rate:

$$\frac{\max_{\sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2} |g_1(\sigma_1, \sigma_2)| \sqrt{M_{\Sigma_1}^2 2^{d_{\Sigma_1}+2} M_{\Sigma_2}^2 2^{d_{\Sigma_2}+2}} \max_{I \in \mathscr{I}} \log |A_I|}{\varepsilon}.$$

A.3 EGT described as a tree traversal

Here we explain how to implement the Prox operation when using the dilated entropy function, as well as how to compute smoothed best responses, i.e. $x_{\mu_1}(y)$ or $y_{\mu_2}(x)$ in Algorithm 5. Throughout we will present algorithms for computing everything from the perspective of a player trying to minimize their opponent's utility, rather than maximize their own.

First, given y^t, the gradient for Player 1 is Ay^t where A is the payoff matrix for Player 2. This can be implemented as follows: create an all-zero vector g of dimension $|\Sigma_1|$. Traverse the game tree, and for each leaf z add $\pi_0(z) y^t[\sigma_2(z)] u_2(z)$ at the entry in g corresponding to $\sigma_1(z)$.

Pseudocode for computing a smoothed best response is given in Algorithm 3. This gives an algorithm for using the dilated entropy function with the negative entropy plus a constant term at each simplex Δ_n: $\sum_x x \log(x) + \log(n)$. By adding $\log(n)$ we ensure that the function is never negative; it is zero at $x_i = \frac{1}{n}$ for all i. Since the constant does not change the second-order derivatives of the dilated entropy function we retain the same strong convexity properties.

The smoothed best response implementation given here modifies the gradient g in place. Thus it is important that g is not used to represent the gradient after a call to the function. However, the modified g can be useful because the entry in g corresponding to the empty sequence then contains the value of the smoothed best response function, which is needed for verifying e.g. the excessive gap condition.

ALGORITHM 3: SmoothedBR

input : gradient g, smoothing parameter μ_1, simplex weights β_I
output: smoothed best response x in sequence form, modified g
`// bottom-up traversal over infosets`
for $I \in \mathscr{I}_1$ **do**
  $w = \mu_1 \beta_I$;
  `// for numerical accuracy we perform an offset`
  $offset = \min(g[I_{start}, I_{end}])$;
  `// set x to smoothed best response`
  $x[I_{start}, I_{end}] \propto \exp(-(g[I_{start}, I_{end}] - offset)/w)$;
  $b = \max(x[I_{start}, I_{end}])$;
  $b\_index =$ index of $\max(x[I_{start}, I_{end}])$;
  `// propagate smoothed best response value at` I
  $g[seq(I)] = g[b\_index] + w(\log(b) + \log(|A_I|))$;
`// At the end, convert x to sequence form`

The algorithm for smoothed best response calculation takes as input a vector g which is usually gradient at the current iteration. This vector g is of length equal to the number of sequences for the player. We assume that the sequences are ordered so that $g[I_{start}, I_{end}]$ denotes the subset of g corresponding to entries for all the sequences that have their last action taken at I. Note that in setting the value of of the indices in x corresponding to I, $x[I_{start}, I_{end}]$, we assume that exp is an index-wise exponential operator and $offset$ is subtracted from each entry.

Given our implementation for smoothed best responses above, the computation of the proximal operator $\text{Prox}_x(g)$ can be performed easily: it is simply a smoothed best response where we shift the gradient g by $\nabla\omega(x)$, where x is the point that we use as the prox center. The following algorithm shifts g in place:

ALGORITHM 4: ProxCenterGradient

input : prox-center x, gradient g, smoothing parameter μ_1, simplex weights β_I
output: smoothed best response x in behavioral form, modified g
`// bottom-up traversal over infosets`
for $I \in \mathscr{I}_1$ **do**
  `// shift` g `at the information set`
  $g[I_{start}, I_{end}] = g[I_{start}, I_{end}] - \mu_1 \beta_I (1 + \log(\frac{x_i}{x_{p_I}}))$;
  `// shift` g `at the parent of` I `by the value at` I
  $g[seq(I)] = g[seq(I)] + \mu_1 \beta_I (1 - \log(|A_I|))$

Note that this implementation of the prox mapping does not give the actual value of the objective, only the strategy vector that minimizes the objective. For smoothed best response the objective could be read off the entry of the modified g at the empty sequence, but it does not hold the correct value for the prox mapping. Unlike for smoothed best response, none of our EGT variants rely on the prox objective.

Once these primitives have been implemented, the high-level steps of Algorithms 1 and 5 are easy to implement. First μ_1 and μ_2 are set to appropriate initial values (for example via the theory or the μ-fitting approach that we use), and initial sequence-form strategies x^0, y^0 are computed for Players 1 and 2 using the above procedures. Then, we just take repeated alternating steps for Players 1 and 2, where the stepsize can either be set to $\frac{2}{t+3}$, or chosen aggressively via heuristics. The most powerful stepsize heuristic is checking whether the excessive gap condition $\bar{\phi}_{\mu_2}(x^t) \leq \bar{\phi}_{\mu_1}(y^t)$ is maintained after every iteration, and then decreasing τ and redoing the most recent step when that condition fails. Given an implementation of smoothed best response, the excessive gap value can be computed as the sum of the smoothed best response values (this works because smoothed best response was implemented to give the value for each player when

they are trying to minimize their opponent's utility). The Step algorithm with these primitives is shown below (assuming that A is the sequence-form payoff matrix for Player 2 and β_1, β_2 are the information-set weights in the dilated entropy for the players):

ALGORITHM 5: StepEFG

input : μ_1, μ_2, x, y, τ
output: μ_1^+, x_+, y_+
// $\hat{x} = (1-\tau)x + \tau x_{\mu_1}(y)$
$g = Ay$;
$x_{\mu_1}(y) = \text{SmoothedBR}(g, \mu_1, \beta_1)$;
$\hat{x} = (1-\tau)x + \tau x_{\mu_1}(y)$;
// $y_+ = (1-\tau)y + \tau y_{\mu_2}(\hat{x})$
$g = -A\hat{x}$;
$y_{\mu_2}(\hat{x}) = \text{SmoothedBR}(g, \mu_2, \beta_2)$;
$y_+ = (1-\tau)y + \tau y_{\mu_2}(\hat{x})$;
// $\tilde{x} = \text{Prox}_{x_{\mu_1}(y)}\left(\frac{\tau}{(1-\tau)\mu_1}\nabla\overline{\phi}_{\mu_2}(\hat{x})\right)$
$g = Ay_{\mu_2}(\hat{x})$;
$\text{ProxCenterGradient}(x_{\mu_1}(y), g, \mu_1, \beta_1)$;
$\tilde{x} = \text{SmoothedBR}(g, \mu_1, \beta_1)$;
// update x_+, u_1^+
$x_+ = (1-\tau)x + \tau\tilde{x}$;
$\mu_1^+ = (1-\tau)\mu_1$;

Finally the EGT algorithm is straightforward as it just iterates calls to StepEFG. The initial points can be computed via SmoothedBR and ProxCenterGradient just as in StepEFG.