A Practical No-Linear-Regret Algorithm for Convex Games

Sue Ann Hong
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
sahong@cs.cmu.edu

Abstract

For convex games, connections between playing by no-regret algorithms and playing equilibrium strategies have previously been made for Φ-regret, a generalization of external regret [5]. In particular, Gordon et al. present a no-Φ-regret algorithm for several different classes of transformations Φ [4]. In this paper, we instantiate the algorithm for the class of linear transformations using a variety of optimization techniques and give experimental results on several games including Indian poker, a simple but substantially large-scale variant of poker. Our results show that both no-external-regret and no-linear-regret algorithms can achieve better regret performances than what the current theory guarantees. To the best of our knowledge, this is the first work empirically demonstrating the benefits of a no-Φ-regret algorithm for general convex games where Φ is stronger than external.

1 Introduction

A multiplayer general-sum game is a way to represent decision problems in a multiagent setting. Here we consider repeated *convex games*, a superset of matrix games and extensive-form games. In the repeated setting, the goal of each player in a game is to minimize his loss over a sequence of instantiations of of the game. In a convex game, a player's loss can be written as a convex function of his *action* as well as other players' actions. In our setting, we assume that after each round a player observes his loss function l_t defined by other players' actions in the past round, but he does not know other players' feasible action regions, only his own feasible space A. Hence the player must learn a course of actions based on their past actions and loss functions. Note that games often involve the bandit setting where a player does not observe the loss function but the actual loss is observed; we do not address the bandit setting in this paper but the no-regret algorithms presented here can be extended without significant changes to the no-regret guarantees.

One way to minimize loss is to play according to a no-Φ-regret learning algorithm that uses information from past rounds of play to determine the action. We define Φ-regret as the difference in loss incurred by the player's past actions a_t at each round t, compared to loss that would have occurred if the player used an alternate sequence of actions $a'_t = \phi(a_t)$ where $\phi : A \mapsto A$, a mapping from an action to an action, is any element of the set Φ. For example, we will focus on the set of linear mappings as Φ in the following sections, which induces no *linear regret*. Following Gordon et al. [4], we write Φ-regret as

$$\rho_T^{\Phi} = \sup_{\phi \in \Phi} \sum_{t=1}^{T} (l_t(a_t) - l_t(\phi(a)))$$

A player achieves no Φ-regret if

$$\sum_{t=1}^{T} l_t(a_t) \leq \sum_{t=1}^{T} l_t(\phi(a_t) + g(T, A, L, \Phi) \forall \phi \in \Phi, t \geq 1$$

where $g(T, A, L, \Phi)$ is sublinear $o(T)$ for any fixed A, L, and Φ.

An additional desirable property of such no-regret algorithm is that they yield equilibrium strategies, as shown in [5]. Intuitively, no-Φ-regret implies that the player has no incentives to have played different action that he has played in the past, at least with respect to possible deviations defined within Φ. One may consider a player with access to only enough computational power to play by a no-Φ-regret algorithm as having bounded rationality. For example, in the case of *swap regret*, which is defined for the broadest class Φ - all possible mappings from A to itself - players using no-swap-regret algorithm end up in a correlated equilibrium. However, a no-swap-regret algorithm may need to consider all possible mappings from A to A which may be infeasible. For example, Stoltz and Lugosi [2] showed the existence of a no-swap-regret algorithm, but their proof emits an algorithm exponential in both time and space with the number of rounds. Hence a player may resort to a weaker no-regret algorithm such as a no-linear-regret algorithm where Φ is the set of linear transformations.

A well-known notion of regret is *external regret*, which is regret compared to a single fixed action for all rounds. Here Φ is the set of functions that map every action to a single action. Hence we can write external regret as:

$$\rho_T^{\Phi} = \sup_{a \in A} \sum_{t=1}^{T} (l_t(a_t) - l_t(a))$$

Many efficient no-external-regret algorithms are available, such as Follow the Perturbed leader [6], GIGA [7], and Lagrangian Hedging [3]. However, no external regret does not guarantee an equilibrium state for the player in general-sum games. Hence we would like to achieve a stronger notion of Φ-regret that guarantees an equilibrium state for the player with bounded rationality. Such no-Φ-regret algorithms exist but until recently they have been impractical, with time complexity exponential in the number of corners of Φ. Gordon et al. [4] introduce a no-Φ-regret algorithm for Φ as broad as the set of finite-element transformations. The algorithm incurs running time polynomial in the dimension of the action set A and the size of the finite-element mesh that covers Φ. However, for simpler transformation spaces such as linear transforms, the dependency on the size of the mesh disappears.

Our goal in this work to show 1. such a no-Φ-regret algorithm is practical and scalable and 2. empirical performance matches the expectations given by the theoretical guarantees. We also compare the performance from using a no-external-regret algorithm to that from using a no-linear-regret algorithm. Our results suggest that many optimization tricks are required to make the algorithm scalable and that empirical performance can exceed the known theoretical guarantees. However, the more computationally expensive no-linear-regret algorithm does not always yield lower regret values for some games; to study why would be an interesting direction for future work.

```
Given: Action space $A$, transformation set $\Phi$.
    1. Pick $\phi_1 \in \Phi$ arbitrarily.
    2. For $t = 1, ..., T$:
        (a) Find an approximate fixed point $a_t$ of $\phi_t$ s.t. $||\phi_t(a_t) - a_t||_A \leq \epsilon_t = 1/\sqrt{t}$.
        (b) Play $a_t$ and receive loss function $l_t$; incur loss $l_t(a_t)$.
        (c) Define "fictitious" loss for $m_t : \Phi \mapsto \mathbb{R}$ by $m_t(\phi) = l_t(\phi(a_t))$ for $\phi \in \Phi$.
        (d) Use a no-external-regret algorithm on $m_t$ and receive a new transformation $\phi_{t+1} \in \Phi$.
```

Figure 1: A general no-Φ-regret algorithm

2 Algorithm

In this section we describe the general no-Φ-regret algorithm from Gordon et al. [4], as well as a linear version of the algorithm used in our experiments.

2.1 A No-Φ-Regret Algorithm

Figure 2.1 shows the general case no-Φ-regret algorithm from Gordon et al. [4]. Note that the algorithm requires two subroutines: a no-external-regret algorithm to be used in the transformation space, and a (approximate) fixed point calculation method for any transformation in Φ. Different types of regret will require different subroutines based on the representation and the size of Φ.

Gordon et al. [4] show that this algorithm achieves no Φ-regret under bounded norm conditions for actions $a \in A$ and the gradient of the loss function $l(a)$. They also briefly discuss specific cases of the algorithm for linear, finite-element, and extensive-form regrets, but do not provide experimental evidence that the algorithms are practical for real-sized games or that the stronger regret guarantee in fact yields more favorable losses.

2.2 A No-Linear-Regret Algorithm

In the linear-regret instantiation of the general algorithm, we employ greedy projection (shown in Figure 2.2) as our no-external-regret algorithm in the transformation space. The most expensive step is the projection of the transformation matrix onto the feasible space Φ; it may take a very large set of constraints to define the space of mappings from the action space A to itself, as many as the number of corners or the number of faces in A. Hence we perform constraint generation on top of the quadratic program in the projection step. Currently we use the dual formulation while still hoping that the number of constraints does not grow too large. Even this formulation yields very large quadratic programs for slightly bigger games. We do not have an implementation that works for Indian Poker, which has a 96-dimensional action space and on the order of $10^9 - 10^{11}$ corners. A few tricks may be used to overcome this computational issue. First, for better constraint generation, we can generate constraints with *margins* and relax those margins as the program becomes infeasible. Also, we can limit the number of Newton-step iterations in the quadratic program solver. Also, an approximate version of the no-linear-regret algorithm is possible; if we do not obtain exactly the transformation from the no-external-regret algorithm but an approximate one, we can project its fixed point onto the feasible action set. This process will still yield an approximate fixed point of the no-external-regret transformation, though with a looser bound. Also, we have not analyzed whether we can construct such a bound in terms of violation of constraints in the projection step for the transformation. Empirically it does not seem that constraint generation would stop very quickly after generating just a

Given: Action space A, loss function l, current transformation matrix M, round number t.
Returns: A feasible matrix M' closest to M with respect to the Frobenius norm.

1. Compute gradient ∂l of the loss function l.
2. Step in the direction of the negative gradient by $\eta = c/t$: $M' = M - \eta * \partial l$. c is some constant.
3. Until no constraints are violated
   (a) Find inequality constraints by finding $x \in A$ that violate $E * M'_i * x \geq d$ where M'_i is a row of M' and $E * x \geq d$ define inequality constraints for the action set A.
   (b) Project M' onto the feasible region defined by equality constraints derived from the null-space and a feasible point of A as well as the inequality constraints found in the previous step, by minimizing $||M'' - M||_F^2$. Set $M' = M''$.

Figure 2: Greedy projection with constraint generation

handful of constraints.

3 Experimental Results

In this section we describe the games used for the experiments and compare the performance of our no-linear-regret algorithm to that of a no-external-regret in terms of swap, linear, and external regret on each game. We used greedy projection as the no-external-regret algorithm, and show results with the best gradient step size found. For the no-linear-regret algorithm we used gradient step sizes .00001/(the round number) and found the performance to be insensitive to the step size[1]. In all cases except for Indian Poker, the regret values were averaged over each player starting the first round with a different strategy from the set of corner strategies. Every player uses the exact same algorithm; we did not consider for example no-external-regret players playing against no-linear-regret players.

3.1 Job Market Game

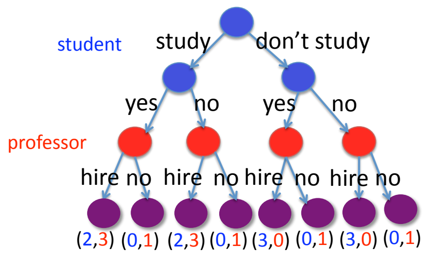

Figure 3: The job market game game tree

We studied the version of the job market game introduced by Forges and Von Stengel [1], whose the game tree is shown in Figure 3. It is a sequential 2-player general-sum extensive-form game where players' strategies can be represented by sequence weights. The student player first decides whether to study or not, then answers yes or no to an interview question. Then the professor decides based on the student's answer to hire the student or not. The Nash equilibrium of the game is for the student to never study and randomly guess the answer to the interview question. However, a correlated equilibrium can be defined if a

[1]We looked at values from .000001 to 100 and did not see much difference in regret values.

moderator tells the student the correct answer to the interview question with high probability if the student studies.

We use sequence weights represent actions as it can lead be exponentially more compact than the game-tree representation when the game involves incomplete information, since the any nodes of the game tree that the player cannot identify as distinct are collapsed into one node in the sequence tree. The space of actions is immediately defined from the sequence weights; each weight must be positive, and the sum of sequence weights associated with a node must add up to the closest ancestor edge of the node. Note that the sequence weight representation also emits a loss function linear with respect to the player's actions.

Figure 4 shows various regret values at each round of the game. The no-external-regret algorithm performs very well for this game and even linear and swap regret decrease almost as fast as external regret, even though there is no known theoretical guarantee for this behavior. It is not surprising given this observation that the no-linear-regret algorithm does not help. However, we do not yet understand why the no-linear-regret algorithm results in worse regret values in this case. The no-linear-regret algorithm was not sensitive to the choice of gradient step size whereas Greedy projection, the no-external-regret algorithm was and we used gradient step sizes 50/(the round number). The average running time for 200 rounds of the no-external-regret algorithm was 1.57 seconds and that for the no-linear algorithm was 37.00 seconds.

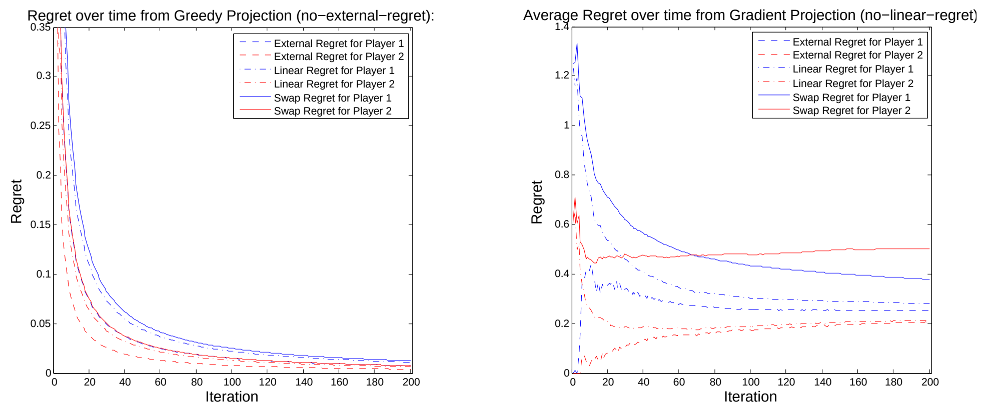

Figure 4: Job Market Game (Left) Regret over rounds for no-external-regret players (Right) Regret over rounds for no-linear-regret players

3.2 The Shapley Game

The Shapley game is a simple variant of Rock-Paper-Scissors where the two players, instead of receiving payoffs of 1/2 each when they tie, receive 0. This modification makes it a general-sum game instead of a constant-sum game, hence a no-external-regret algorithm may not yield an equilibrium strategy. Depending on the starting strategies for each player, the no-external-regret algorithm led the players to the Nash equilibrium of playing each move 1/3 of the time or oscillating between different strategies, not necessarily converging to one. The no-linear-regret algorithm always resulted in both players converging to the (1/3, 1/3, 1/3) strategy. Figure 5 shows the comparison between no-external and no-linear regret algorithms for the Shapley Game. The no-external-regret algorithm does not achieve swap regret in the 200 iterations shown though it does achieve no linear regret. The no-linear-regret algorithm on the other hand achieves much lower swap regret very fast, even though no linear regret does not guarantee no swap regret. For no-external-regret we used

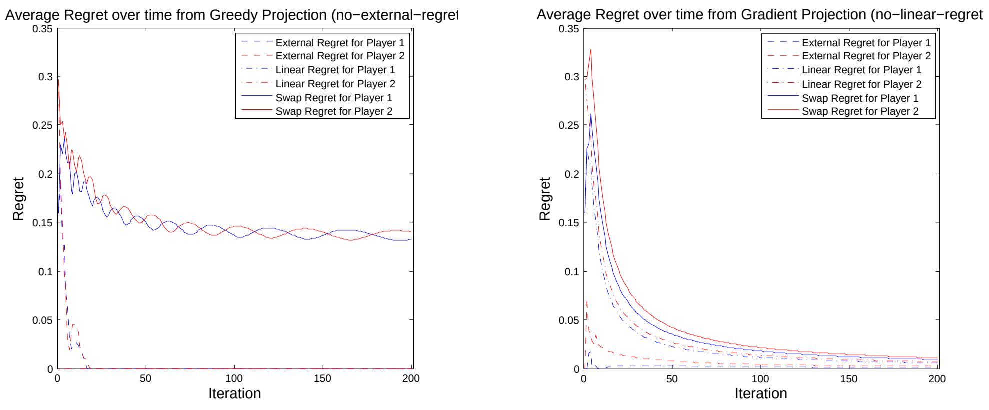

Figure 5: The Shapley Game (Left) Regret over rounds for no-external-regret players (Right) Regret over rounds for no-linear-regret players

gradient step sizes 10/(the round number), and the performance did vary quite a bit depending on the step size, sometimes resulting in high external regret and linear regret. The average running time for 200 rounds of the no-external-regret algorithm was 1.17 seconds and that for the no-linear algorithm was 10.35 seconds.

3.3 A Bayes-Shapley Game

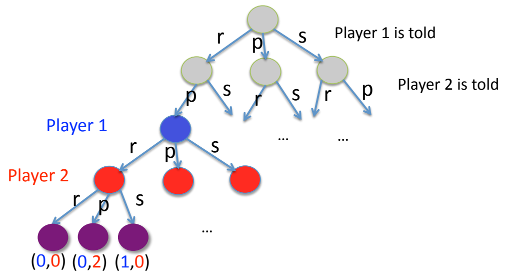

Figure 6: The Bayes-Shapley game game tree

We also considered a sequential Bayes version of the Shapley game where each player is first secretly told a unique move [2] to make and then decides on the move; the player receives double the normal payoff if she follows the move she was told in the beginning. The game tree partially is shown in Figure 6. There are 54 leaves in the game tree and 9 sequence weights for each player.

Figure 7 shows results from simulating no-external and no-linear regret players. Here, the no-linear-regret player ends up with huge linear regret, which theoretically should not happen; the average regret is growing (seemingly) linearly, where the theory dictates the cumulative regret should grow sublinearly.

For no-external-regret we used gradient step sizes 10/(the round number), though we have not tested each algorithm's sensitivity to the step size for this game. The average running time for 200 rounds of the no-external-regret algorithm was 2.06 seconds and that for the no-linear algorithm was 38.34 seconds.

[2]Each player is told a different move from the other.

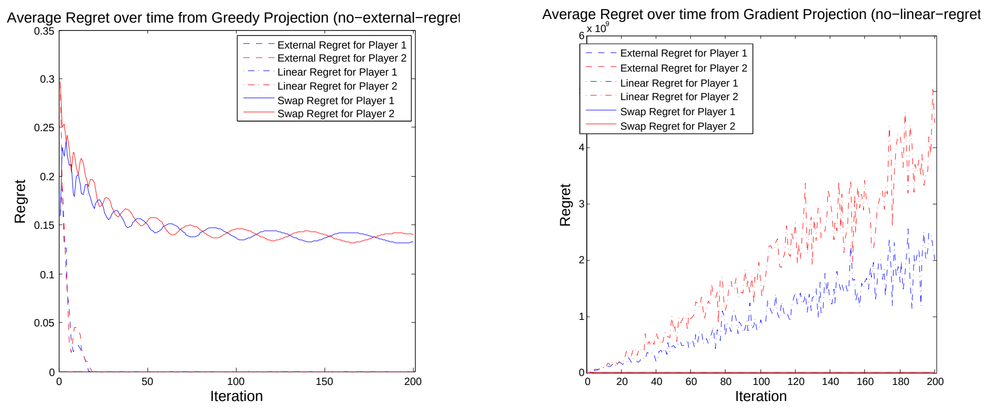

Figure 7: The Bayes-Shapley Game (Left) Regret over rounds for no-external-regret players (Right) Regret over rounds for no-linear-regret players

3.4 Indian Poker

In 3-player 4-card Indian poker, each person is dealt a card which he does not observe; instead he sticks it on his forehead. Hence each player observes other players' cards, and the remaining undealt card is hidden. We consider one round of betting with \$1 ante and a cap of \$1 (no raise), which leads to the maximum betting sequence of five decisions over three players. The game tree is omitted for spatial reasons.

Again we use the sequence-weights representation of the action space, which is especially useful for a larger game like Indian Poker. In indian poker, sequence weights (hence the actions) are 96-dimensional vectors. Unfortunately the number of corners of the action space A is on the order of 10^9 to 10^{11} depending on the player, which makes representations of Φ that depend on the corners of A inconvenient.

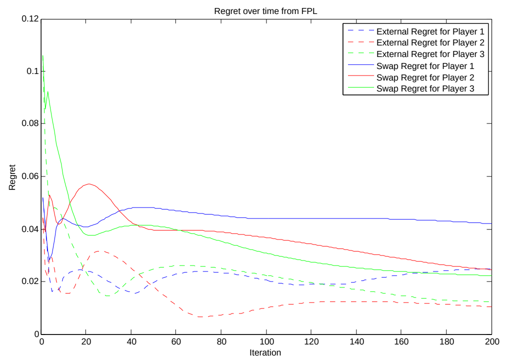

Figure 8: Indian Poker. Regret over rounds for no-external-regret players

Since we do not have an implementation that works for a game of this size, we present the results from greedy projection, our no-external-regret algorithm. For these simulated rounds, each player starts the first round with a corner action and every player in the game uses the exact same algorithm. Figure 8 shows swap and external regret over the rounds of game for all three players using greedy projection. We set $c = 50$ for the gradient step size as in the no-linear-algorithm of Figure 2.2. For the first 200 rounds swap regret does not seem to be decreasing too fast, but neither does external regret, so it is difficult to say what will happen with a no-linear-regret algorithm.

Note that computing linear regret requires finding the best linear transformation given all cost functions in the past. For extensive-form games where strategies are represented by sequence weights, the cost function is linear with respect to actions, hence we can formulate the problem as a linear program. However, the space of Φ requires a huge number of constraints (recall the number of corners in Indian poker is on the order of 10^9), so we again use constraint generation in conjunction with the linear program. While we only need a linear program, not a quadratic program as in the case of projection, this step suffers from the same constraint generation problems as our projection procedure.

4 Conclusion

In this paper we presented the first empirical results of using a no-linear-regret algorithm in repeated convex games. We examined four different general-sum games: a 2-player matrix game, two 2-player extensive-form games, and a larger 3-player extensive-form game. Overall, we found greedy projection (no-external-regret) to be much more sensitive to gradient step size than gradient projection (no-internal-regret) which seemed almost invariant to the step size. Linear regret was easily achieved by the no-external-regret algorithm in our experiments, which theoretically makes the no-linear-regret algorithm less interesting since it only guarantees no linear regret. However, we found that it is possible to achieve no swap regret using our no-linear-regret algorithm at least in one case - the Shapley game.

There are several directions for future work. First, it would be interesting to examine the loss or regret of players using different algorithms (e.g. one player uses no-linear and others use no-external). We would also like to extend the no-linear-regret algorithm to a larger set of transformations such as those defined by two linear transformations each over a subset of the action space (a very small subset of finite-element transformations). Another direction would be to formalize the conditions under which no-linear-regret or no-finite-element-regret algorithms can help improve regret performances compared to a no-external-regret algorithm.

References

[1] Francoise Forges and Bernhard von Stengel. Computationally efficient coordination in game trees. In *Technical Report LSE-CDAM-2002-02, London School of Economics and Political Science, Centre for Discrete and Applicable Mathematics*, 2002.

[2] Stolz G. and Lugosi G. Learning correlated equilibria in games with compact sets of strategies. In *Games and Economic Behavior, 59*, 2007.

[3] Geoff Gordon. No-regret algorithms for online convex programs. In *Proceedings of the Neural Information Processing Systems*, 2006.

[4] Geoff Gordon, Amy Greenwald, Casey Marks, and Martin Zinkevich. No-regret learning in convex games. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[5] Amy Greenwald and Amir Jafari. A general class of no-regret learning algorithms and game-theoretic equilibria. In *The Proceedings of the 2003 Computational Learning Theory Conference*, 2003.

[6] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. In *Proceedings of the 16th Annual Conference on Learning Theory*, 2003.

[7] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.