SOLVING MULTI-AGENT ZERO-SUM GAMES WITH MIRROR DESCENT

A PREPRINT

Rohan Sinha[*]
Department of Aeronautics and Astronautics
Stanford University
rhnsinha@stanford.edu

1 Introduction

To reliably deploy the next generation of autonomous robotic systems in unstructured open-world environments, they must safely interact with other agents such as humans and other robots. In many of these settings, such as when an autonomous vehicle must merge into traffic on a highway onramp, automated systems need to compete with other agents to force their objective. Therefore, this requires the development of strategies that intelligently reason about the effect that the decisions made by an automated robot have on the actions of the agents in its environment. Recent work advocates a game-theoretic approach [2, 3], where opposing agents are modelled as rational with respect to a cost function to construct policies that anticipate how opponents will react. The agents are said to be at a Nash-equilibrium if it is in none of the agents' interest to change their behavior, given that the others keep their policies the same [4]. Since Nash-equilibria result in a stable interaction, they are often desirable policies to search for in a competitive setting.

However, solving for a Nash equilibrium is generally intractable [4] and ambiguity arises when multiple equilibria exist to the game. Only a handful of classical results exist where a unique solution can be found efficiently, all of which concern two-player zero-sum games in either the static [5], linear quadratic dynamic [2], or stochastic dynamic case [6], since this results in adversarial behavior that convex duality and minimax theorems allow us to solve. However, recent work proved that a Nash equilibrium can be efficiently found for a class of static zero-sum multi-agent ($n > 2$) games using linear programming [7].

In my research, we are interested in extending these static results to the stochastic dynamic setting (also known as a stochastic game or multi-agent MDP) to study strategic multi-agent interactions in autonomous systems and to disambiguate effective policies when many Nash equilibria exist. A major roadblock is the fact that the computational complexity of multi-agent approaches inevitably scales poorly in the number of agents. For example, for a multi-agent MDP with 5 agents that represent distinct robots that can move on a 10 by 10 grid, applying a value iteration algorithm requires solving $\mathcal{O}((10 \times 10)^5)$ LP subproblems at each iteration. The goal of this project is to take advantage of the structure in the linear programs encountered in game theory to develop fast solvers to mitigate computational issues. In particular, we exploit the structure in the games considered in [7] and apply the mirror descent algorithm with a Polyak step-size to rapidly find Nash-equilibria. We provide some basic analysis on stepsize selection, and validate the convergence of our algorithm and compare its performance with a generic LP solver using simulated experiments. In addition, we contrast the benefit of our approach with a method inspired by the Analytical Centre Cutting Plane (ACCPM) algorithm. Even though a cutting plane algorithm requires fewer iterations to converge, the computational cost of an iteration increases greatly with the problem size compared to our first order method. Finally, we examine distributed algorithms like the consensus Alternating Direction Method of Multipliers (ADMM) applied to the multi-agent problems we consider, and draw connections to typical Iterative Best Response (IBR) algorithms that are often considered in the literature [3, 2].

2 Background: Zero-Sum Multi-Agent Games

In this section we review the results of [7] to define the problem we solve in this work and establish some of its properties. A poly-matrix game consists of n agents that each play an action at the same time. The actions selected by the players determine the outcome of the game as the payoffs received by each player. Each player tries to maximize

[*]This work is related to research on urban autonomous driving. For some simulation videos of competitive urban driving scenarios in CARLA [1], see `https://drive.google.com/drive/folders/1qT_5MDmoStbdvXBmkbR_0PUf4hnPbHYm?usp=sharing`.

their payoff with complete knowledge of the incentive structure of the other agents. A zero-sum poly-matrix game differs from general-sum multi-agent games because the interactions between the agents, in the form of the payoff functions, are taken as pairwise and net zero. To define the game, we use the following notation:

- $\mathcal{S}_i$ is the discrete action set for agent $i \in \{1, \dots, n\}$.
- $p^{ij} : \mathcal{S}_i \times \mathcal{S}_j \mapsto \mathbb{R}$ is the payoff function for agent i associated with the interaction between agents i, j. Although it is not required for $p^{ij}(\cdot) = -p^{ji}(\cdot)$, it does make the game zero-sum if true for all agents.
- We call a collection of actions for each agent a strategy profile $\bar{s} = (s_1, \dots, s_n) \in \prod_{i=1}^n \mathcal{S}_i$.
- the payoff for agent i under the strategy profile $\bar{s}$ is $p_i(\bar{s}) = \sum_{j \neq i} p^{ij}(s_i, s_j)$. We write payoffs as functions of the profiles of each agent for convenience.
- The game is zero sum if and only if $\sum_{i=1}^n p_i(\bar{s}) = 0$ for any strategy profile $\bar{s}$.
- We write stochastic, often referred to as mixed, strategies for agent i as $x_i \in \mathcal{P}^{|\mathcal{S}_i|}$ with $\mathcal{P}^k = \{x \in \mathbb{R}^k : x \geq 0,\ \mathbf{1}^\intercal x = 1\}$ as the k-dimensional probability simplex. A stochastic strategy profile $x = [x_1^\intercal, \dots, x_n^\intercal]^\intercal$ is valid if each $x_i \in \mathcal{P}^{|\mathcal{S}_i|}$. We then say $x \in \Delta$. Note Δ is a convex set.
- For a mixed profile x, we denote the profile of all the agents but i as $x_{-i} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$. With a slight abuse of notation, we write the expected payoff for an agent i under a mixed profile x using the payoff function: $p_i(x) := p_i(x_i, x_{-i}) := \mathbb{E}_{\bar{s} \sim x}(p_i(\bar{s}))$. We also write the expected payoff for agent i if agent i plays a pure strategy $s \in \mathcal{S}_i$ and the other agents play the mixed profile x_{-i} as $p_i(s, x_{-i}) := \mathbb{E}_{s_j \sim x_j, j \neq i}(p_i((s_1, \dots, s_n)))$.

As the agents must compete with each other, there is no single strategy profile that optimizes the performance of all the agents. Instead, we must pursue a stable interaction for which it is in none of the agents' respective interest to defect:

Definition 1. *(Nash Equilibrium [4]) A mixed strategy profile $x^* \in \Delta$ for a multi-agent game is a Nash equilibrium if and only if*

$$x_i^* \in \underset{x_i \in \mathcal{P}^{|\mathcal{S}_i|}}{\arg\max}\, p_i(x_i, x_{-i}^*) \quad \forall i \in \{1, \dots, n\} \tag{1}$$

To interpret definition 1, consider a game between two players: Bob and Alice. Definition 1 states that a mixed strategy for Bob is Nash-optimal if and only if information about Alice's strategy cannot increase his expected payoff and vice versa. For many agents, this means that the interaction is stable: None of the agents can unilaterally change their strategy and receive a higher payoff. For many games, like rock-paper-scissors, there are no deterministic ("pure") strategy profiles that form a Nash equilibrium. Instead, it is a classic result in game theory that a Nash equilibrium must exist in the space of mixed (stochastic) strategy profiles.

Theorem 1. *(Nash's Theorem [4]) For any general-sum multi-agent game, there exists a mixed strategy profile that is a Nash equilibrium.*

We are now ready to quote the main result of [7] and define the problem we develop a solver for. For the rest of this report, we will restrict ourselves to zero-sum polymatrix games.

Theorem 2. *(Polymatrix Game Theorem [7]) Consider a zero-sum polymatrix game, and let $x^* \in \Delta$, $w^* \in \mathbb{R}^n$ be the solution of the associated LP (2). Then $\sum_{i=1}^n w_i^* = 0$ and x^* is a Nash equilibrium.*

$$\begin{aligned} \min_{w,x} \quad & \sum_{i=1}^n w_i \\ s.t. \quad & w_i \geq p_i(s, x_{-i})\ \forall s \in \mathcal{S}_i,\ i \in [1, n] \\ & x \in \Delta \end{aligned} \tag{2}$$

3 Proposed Approach

To see why problem (2) is an LP, we define the payoff matrices A^{ij} as matrices with elements such that $A^{ij}_{kl} = p^{ij}(k, l)$ for $k, l \in \mathcal{S}_i, \mathcal{S}_j$. Then, for a mixed strategy profile $x \in \Delta$,

$$\mathbb{E}_{\bar{s} \sim x}(p^{ij}(s_i, s_j)) = \sum_{k \in \mathcal{S}_i} \sum_{l \in \mathcal{S}_j} p^{ij}(k, l) x_i^k x_j^l = x_i^\intercal A^{ij} x_j \tag{3}$$

It follows that $p_i(x) = \sum_{i \neq j} x_i^\intercal A^{ij} x_j$. Therefore, if we define the matrix A as

$$A = \begin{bmatrix} A^1 \\ \vdots \\ A^n \end{bmatrix} = \begin{bmatrix} 0 & A^{12} & \dots & A^{1n} \\ \vdots & \ddots & \ddots & \vdots \\ A^{n1} & A^{n2} & \dots & 0 \end{bmatrix} \tag{4}$$

then $x^\intercal A x = \sum_i p_i(x) = 0$. In addition, we recognize that (2) is an optimization over a pointwise maximum written in epigraph form, so we eliminate w and rewrite the LP (2) as

$$\begin{aligned} \min_x \quad & \sum_{i=1}^{n} \max_{j \in |\mathcal{S}_i|} \{A_j^i x\} \equiv f(x) \\ s.t. \quad & x \in \Delta \end{aligned} \tag{5}$$

where A_j^i indicates the j-th row of A^i. Problem (5) has a convex objective and a single simplex-like constraint.

Remark 1. *The LP (5) makes it explicit why the interactions in the polymatrix game need to be pairwise: otherwise the inner maximization would be over products of decision variables resulting in a nonconvex problem.*

3.1 Mirror Descent with Polyak Stepsize

We know apply the Mirror Descent algorithm to the reformulated LP (5). From the subgradient calculus rules [8], it follows that

$$g = \sum_{i=1}^{n} A_{j_i^*}^i \in \partial f(x), \tag{6}$$

with $j_i^* \in \arg\max_{j \in |\mathcal{S}_i|} \{A_j^i x\}$. To take advantage of the structure of the simplex-like constraint in (5), we consider the application of the Mirror Descent algorithm for the Bregman divergence associated with the negative entropy function [8]. In this case, this results in the generalized KL divergence associated with $x, y \in \mathbb{R}^m$,

$$D(x, y) = \sum_{i=1}^{m} x_i \log\left(\frac{x_i}{y_i}\right) - (x_i - y_i). \tag{7}$$

Since the simplex constraints in (5) form independent constraints on the strategy profile of each respective agent, the generalized projection in the Mirror Descent algorithm results in independent updates for each agent. We use the notation $g_{i,j}$ to refer the j-th entry of the i-th block of the subgradient $g = [g_1^\intercal, \dots, g_n^\intercal]^\intercal$ associated with agent i for clarity. This gives the Mirror Descent update for agent i as

$$x_i^{(k+1)} = \arg\min_{x \in \mathcal{P}^{|\mathcal{S}_i|}} D(x, x^{(k)}) + \alpha_k x^\intercal g_i^{(k)}. \tag{8}$$

Using the simplex structure, the resulting Mirror Descent update for (5) simplifies to

$$x_{i,j}^{(k+1)} = \frac{x_{i,j}^{(k)} \exp(-\alpha_k g_{i,j}^{(k)})}{\sum_{l=1}^{|\mathcal{S}_i|} x_{i,l}^{(k)} \exp(-\alpha_k g_{i,l}^{(k)})} \tag{9}$$

where $x_{i,j}^{(k+1)}$ indicates the probability of selecting the j-th action for agent i at the $(k+1)$-th iteration [8].

Remark 2. *Since the Mirror Descent updates are independent for each agent, computational speedup can be achieved by parallelizing the update procedure.*

3.2 Stepsize Selection

Finally, we need to decide on a stepsize schedule. Since theorem 2 states that any optimal solution to the LP (2) has 0 objective and theorem 1 states that a Nash equilibrium must exist, we know that (5) is always feasible and that $f(x^*) = f^* = 0$. This motivates application of the optimal Polyak stepsize, given by

$$\alpha_k = \frac{f(x^{(k)}) - f^*}{\|g^{(k)}\|_2^2} = \frac{f(x^{(k)})}{\|g^{(k)}\|_2^2}. \tag{10}$$

We provide some basic qualitative analysis to establish what kind of convergence behavior we should expect from the Polyak stepsize (10) as compared to other standard stepsize schedules. To do this, consider the basic convergence analysis equation that results from the properties of the subgradients [8]:

$$\|x^{(k+1)} - x^\star\|_2^2 \leq \|x^{(k)} - x^\star\| - 2\alpha_k (f(x^k) - f^\star) + \alpha_k^2 \|g^{(k)}\|_2^2 \tag{11}$$

For a Polyak stepsize, this equation simplifies to

$$\|x^{(k+1)} - x^\star\|_2^2 \leq \|x^{(k)} - x^\star\| - \frac{(f(x^k) - f^\star)^2}{\|g^{(k)}\|_2^2}. \tag{12}$$

This establishes an important property of the Polyak stepsize: the distance of the solution to the true optimal solution, $\|x^{(k)} - x^\star\|$, must decrease at each iteration. In contrast, for a regular stepsize schedule, the residual $\alpha_k^2\|g^{(k)}\|_2^2$ makes it possible for the solution iterates to stray further from the solution at any particular iteration. Moreover, convergence rates are typically established using an upper bound on the subgradients $\|g\|_2 \leq G$. In our experiments, we take the entries of the payoff matrices as integers, and so we can expect the ℓ_1 norm of the subgradients to grow at least linearly in the number of agents or actions available to each agent. Since $\|g\|_1 \leq \sqrt{n}\|g\|_2$, this means that the bounds on the performance improvement deteriorate in the problem size for regular stepsizes. Moreover, the faster the stepsizes decay, the smaller the number of iterations without performance improvement will be. This is evident from the standard convergence result (with $\|x^1 - x^\star\| = R$) [8]:

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{R^2 + G^2 \sum_{i=1}^k \alpha_k^2}{\sum_{i=1}^k \alpha_k} \tag{13}$$

However, the Polyak stepsize is not perfect, as the Polyak convergence result simplifies to $k(f_{\text{best}}^{(k)} - f^\star)^2 \leq \sum_{i=1}^k (f^{(k)} - f^\star)^2 \leq R^2G^2$. This implies convergence on the order of $RG/\sqrt{k}$, still showing deteriorating performance in the problem dimension. The Polyak stepsize will therefore perform better significantly better for larger problems, as for a standard stepsize schedule (13) shows that the dependence is quadratic in the problem size, meaning that it may take a long time before the iterates start to improve the solution for large problems. A final method to speedup the convergence of (Polyak) stepsize methods is to filter the subgradients using a constant $\beta \in [0, 1]$ to generate an analogue to a momentum method:

$$s^{(k)} = (1 - \beta)s^{(k-1)} + \beta g^{(k)}. \tag{14}$$

Moreover, if we pick a small β and set the initial filtered subgradient $s^{(1)}$ to zero, it will take a long time for the filtered subgradient to grow in norm. Therefore, we should be able to curb the impact of the problem size on a standard stepsize schedule using heavy filtering.

3.3 Analytic Center Cutting-Plane Method (ACCPM)

We use the Analytic Center Cutting-Plane method [8] as an example approach to compare our Mirror Descent algorithm to. Cutting-Plane methods query a computational oracle at a candidate point x to return a seperating hyperplane that defines a halfspace containing the optimal solution set. This binary refinement typically allows these methods to converge with much fewer iterations than relatively slow subgradient methods. To compute these cuts, we iteratively evaluate the objective and associated subgradient of the objective of (5). The basic subgradient definition gives that for a $g \in \partial f(x)$, $f(z) \geq f(x) + g^\intercal(z - x)$. Moreover, from theorem 2 we know that $f^\star = 0$. Therefore, at a candidate point x, this establishes the *deep cut*

$$g^\intercal z \leq g^\intercal x - f(x) + f^\star = g^\intercal x - f(x). \tag{15}$$

After adding a cutting plane, we compute the analytical center of our collection of cutting planes $\{(a_i, b_i)\}_{i=0}^k$ given as

$$x^{(k+1)} = \arg\min_x \sum_{i=0}^k -\log(b_i - a_i^\intercal x) \tag{16}$$

using an infeasible start Newton Method by clipping negative entries in the log to 1 (see [8]). A challenge is that centering methods like the analytical center typically only perform well when the set of constraints has volume, i.e. that we only have inequality constraints. However, the LP (5) has equality constraints formed by the simplices. We found that adding the equality constraints by converting $\{x : Ax = b\}$ to $\{x : Ax - b \leq 0\} \cap \{x : 0 \leq Ax - b\}$ performed extremely poorly in practice. Instead, we modified the ACCPM algorithm to only use the cutting planes as inequality constraints and project the AC into the simplex at each iteration instead. We then took the initial constraint polytope as $\{x : x \geq 0\} \cap \{\|x\|_\infty \leq 1\} \supset \Delta$. The ACCPM method is somewhat reflective of the tactics used in the standard solvers included in CVX. The number of cutting planes required to narrow the solution down increases greatly in the dimensionality of the decision space. Moreover, the Newton Method used for centering greatly increases in computational cost in the number of decision variables. Hence, we can expect this baseline to perform poorly for large problems with many agents.

3.4 Distributed Alternating Direction Method of Multipliers (ADMM)

Iterative Best Response (IBR) algorithms are typically the method of choice to find Nash equilibria for dynamic games in applied work [2, 3]. In these algorithms, each agent iteratively optimizes their strategy as the best response to the strategies of its opponents at the current iteration. This typically happens by cycling through all the agents in a fixed order. However, for the discrete games that we consider, such algorithms generally won't converge. To see this, observe that the best response set for agent i in the polymatrix games we consider is formed by all $x_i \in \arg\max_{x_i \in \mathcal{P}^{|\mathcal{S}_i|}} \sum_{j \neq i} x_i A^{ij} x_j^{(k)}$. This set includes the standard basis vector associated with $\max_m A^i_m x^{(k)}$, since $A^{ii} = 0$. This implies that there is always a deterministic best response to any given policy profile, even though many games only have stochastic (mixed) profiles for equilibria: For example, for rock-paper-scissors the equilibrium is to select an action uniformly with probability $1/3$. Therefore, deterministic best responses will never converge to a mixed equilibrium.

However, we can tease out a related decomposition structure to the IBR approach from the LP (5). By introducing the subproblem

$$\min_{z_i \in \Delta} \max_j \{A^i_j z_i\} \tag{17}$$

and the global consensus variable x with associated constraint $z_i = x$ for $i = 1, \ldots, n$, we have decomposed the LP (5) into a subproblem associated with the best response of each agent. Instead of cycling through the subproblems, we achieve computational speedups by taking a distributed approach. To do this, we apply the consensus ADMM update equations [8], resulting in the distributed updates

$$z_i^{(k+1)} = \arg\min_{z_i \in \Delta} \max_j \{A^i_j z_i\} + (z_i - x^{(k)})^\intercal y_i^{(k)} + \frac{\rho}{2} \|z_i - x^{(k)}\|_2^2. \tag{18}$$

We did not include the standard averaging step for $x^{(k)}$ and updates of the dual variables $y_i^{(k)}$ here for brevity (see [8]). The benefit of applying the distributed update is that we now only need to optimimze over a single pointwise max in contrast with the LP (5), resulting in a reduced computational load that can be computed in parallel in a distributed fashion. We can view these updates as somewhat of a converse to a standard IBR update: Rather than updating the policy of one agent as a best response to all the other policies, we update the policies of all the other agents to minimize the best response of a single agent (since $A^{ii} = 0$, the profile of agent i is a free variable in (18)). By averaging all the minimizers together, we achieve a global Nash equilibrium. Instead of computing this equilibrium in sequence, i.e. using vanilla dual ascent, we use distributed updates and take advantage of the favorable robustness properties of the consensus ADMM algorithm.

4 Experiments

To test the efficacy of our method, we created a simple benchmark problem and vary its size. To do this, we first fixed the number of agents and the number of actions available to each agent. We then sampled $p^{i,j}(\cdot)$ randomly between $[-5, 5]$ and set $p^{i,j}(\cdot) = -p^{j,i}(\cdot)$ to ensure the game is zero sum. We compared the performance of the Mirror Descent algorithm with the Polyak stepsize (10) to a constant stepsize, a first order stepsize $1/k$, and a square root stepsize $1/\sqrt{k}$.

In our first experiment, we fixed the number of agents to $n = 10$ and compared our method to other stepsizes as the number of actions available increased for each agent. These results are shown in figure 1. An interesting pattern emerges: For a small problem, the $1/k$ stepsize performed best, followed by a slightly slower Polyak stepsize. The square-root stepsize and the constant step clearly performed much slower. For a larger problem, with 100 actions available to each of the agents, the $1/k$ step still outperforms the Polyak step. However, the Polyak step improves the objective immediately, whereas all the other stepsizes do not improve the estimate at first for a significant number of iterations. This effect becomes more pronounced when the agents are given 500 actions to choose from. Both the Polyak and $1/k$ step seem to converge at a similar rate, but the head-start from the Polyak step means it now converges more rapidly. Moreover, the constant stepsize fails to converge. Finally, we test our approach on a large problem with 1000 actions for each agent. Clearly, the Polyak stepsize significantly outperforms the other methods on this problem. Looking at the objective values, we see that the number of iterations required to reach a similar objective (or distance from a nash equilibrium) is similar. We conclude that the Polyak stepsize converges much faster for large problems. These results are in line with the discussion in section 3.2: The Polyak stepsize always improves and regular stepsizes deteriorate in performance as the problem size increases. Moreover, the faster the stepsizes decay, the better the convergence behavior. This is in line with our understanding that quickly decaying stepsizes curb the impact of the large subgradients for large problems.

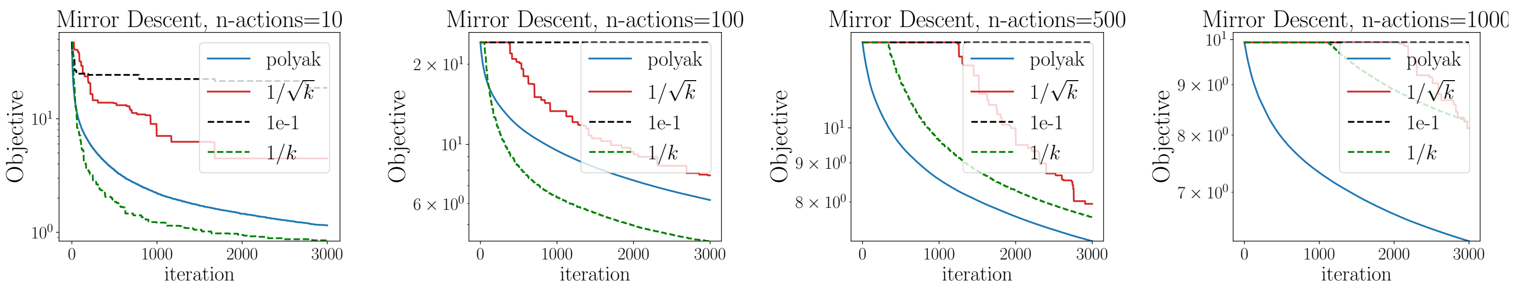

Figure 1: Mirror Descent convergence for various problem sizes. Plotted lines indicate best value seen so far. We emphasize that the optimal objective is 0.

In addition, figure 2 (leftmost 2 plots) show that filtering the subgradients allows standard subgradient methods to start converging faster for a large problem. As the filter becomes slower (smaller β), the performance improves, although we found the Polyak stepsize to still outperform other methods for large problems.

We ran a similar experiment where we kept the number of actions constant and increased the number of agents. The results were similar to figure 1, so we did not include them for brevity. Comparing our method to a naive implementation in CVXPY, we see significant potential for our method: For the larger problems $n = 500, 1000$, our CVXPY implementation took > 15 minutes to solve, whereas it took < 5 minutes to compute 3000 iterations of our Mirror Descent algorithm on a standard laptop. Since it is not a fair comparison to contrast python code with CVX solvers in C++, we also compare against the ACCPM solver. As shown in figure 2(2nd from right), the ACCPM solver converges in a few iterations for a small problem of 10 agents and 10 actions. However, for the larger problems with 1000 agents, the Newton iterations were so slow that it took more than 5 minutes to compute 10 iterations. This clearly underscores the benefit of a first order method on large problems. Finally, we plot the convergence of the consensus ADMM algorithm on the small subproblem in figure 2. As is apparent, the consensus average and the dual objective converge to 0, the true optimal. Notably, it only takes on average 5 seconds to compute the distributed updates using CVXPY for a problem of size 1000 agents with a solver time of .1 seconds. In contrast, it took almost 20 minutes to solve the full problem using CVXPY. This underscores that the ADMM decomposition can result in rapid distributed solution methods.

5 Conclusion and Future Work

We conclude that Mirror Descent with a Polyak stepsize is a promising method to identify Nash-equilibria of large multi-agent games, as qualitative analysis and our experiments show the Polyak stepsize is less affected by problem size. Although such first-order methods typically require many iterations, we see that they vastly outperform cutting-plane methods for problems with large numbers of agents. In addition, we show that a distributed consensus ADMM algorithm vastly reduces the computation time of the subproblems, allowing fast convergence if many distributed processors are available. However, our current implementation in Python is not a fast, packaged solver. To do this, we should consider rewriting our code in a faster language (like C++) and including stopping criteria to return a final solution. It is our expectation that our solution methods will then outperform generic solvers for most problem sizes. The code for this project is available at `https://github.com/GoldeneyeRohan/mirror_descent_polymatrix`, please request access by emailing the author.

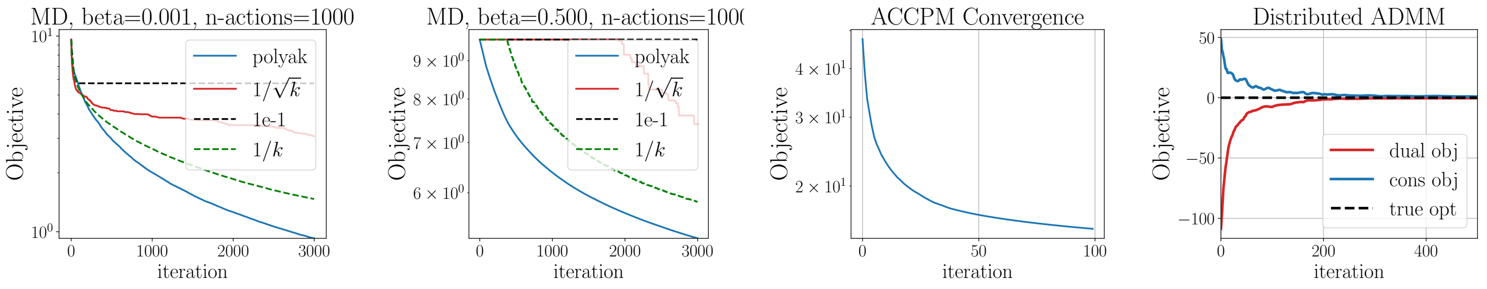

Figure 2: Left:Mirror Descent with filtered subgradients. Center Right: Convergence of ACCPM. Right: Convergence of consensus ADMM, 'dual obj' indicates the dual objective value, 'cons obj' the primal objective value of the consensus variable.

References

[1] A. Dosovitskiy. Carla: An open urban driving simulator.

[2] David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D. Dragan, and Claire J. Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1475–1481, 2020.

[3] Riccardo Spica, Davide Falanga, Eric Cristofalo, Eduardo Montijano, Davide Scaramuzza, and Mac Schwager. A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing. *IEEE Transactions on Robotics*, 36(5), 2020.

[4] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.

[5] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani. *Algorithms*. McGraw-Hill, 2008.

[6] L. S. Shapley. Stochastic Games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, October 1953. Publisher: National Academy of Sciences Section: Mathematics.

[7] Yang Cai, Ozan Candogan, Constantinos Daskalakis, and Christos Papadimitriou. Zero-Sum Polymatrix Games: A Generalization of Minmax. *Mathematics of Operations Research*, 41(2):648–655, January 2016. Publisher: INFORMS.

[8] Mert Pilanci and Stephen Boyd. *Course Notes for EE364B at Stanford University*. Stanford University, 2021.