

SIAG on Optimization Views and News

A Forum for the [SIAM Activity Group on Optimization](#)

Volume 31 Number 1

December 2023

Contents

Articles

First-Order Methods for Linear Programming

Haihao Lu 1

Global Convergence of Policy Gradient Methods in Reinforcement Learning, Games and Control

Shicong Cen, Yuejie Chi.....12

Bulletin

Event announcements..... 22

Chair's Column

Luis Nunes Vicente..... 23

Comments from the Editors

Dmitriy Drusvyatskiy, Matt Menickelly, Pietro Belotti... 24

Articles

First-Order Methods for Linear Programming



Haihao Lu

Booth School of Business

The University of Chicago

haihao.lu@chicagobooth.edu

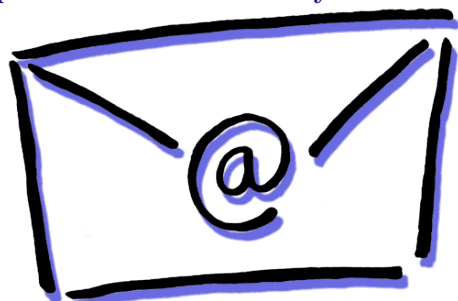
1 Introduction

Linear programming (LP) [26, 17, 15, 51, 50, 33] is a seminal optimization problem that has grown with today's rich and diverse optimization modeling and algorithmic landscape. LP is used in just about every arena of the global economy, including transportation, telecommunications, production and operations scheduling, as well as in support of strategic decision-making [24, 19, 14, 30, 13, 53]. Eugene Lawler is quoted as stating in 1980 that LP "is used to allocate resources, plan production, schedule workers, plan investment portfolios and formulate marketing (and military) strategies. The versatility and economic impact of linear optimization models in today's industrial world is truly awesome."

Since the late 1940s, the exceptional modeling capabilities of LP have spurred extensive investigations into efficient algorithms for solving LP. Presently, the most widely recognized methods for LP are Dantzig's simplex method [15, 16] and interior-point methods [46, 36, 52]. The state-of-the-art commercial LP solvers, which are based on variants of these two methods, are quite mature, and can reliably deliver extremely accurate solutions. However, it is extremely challenging to further scale either of these two algorithms beyond the problem sizes they can currently handle. More specifically, the computational bottlenecks of both methods involve matrix factorization to solve linear equations, which leads to two fundamental challenges as the size of the problem increases:

- In commercial LP solvers, the simplex method utilizes LU-factorization and the interior point method utilizes Cholesky factorization when solving the linear equations. While the constraint matrix is extremely sparse in practice, it is often the case that the factorization is much denser than the constraint matrix. This is the reason why commercial LP solvers require more memory than just storing the LP instance itself, and it may lead to out-of-memory errors when solving large instances.

Are you receiving this by postal mail?
Do you prefer electronic delivery?



Email siagoptnews@lists.mcs.anl.gov

Even when it does fit in memory, the factorization may take a long time for large instances.

- It is highly challenging to take advantage of modern computing architectures, such as GPUs or distributed systems, to solve the linear equations in these two methods. For simplex, multiple cores are generally employed only in the “pricing” step that selects variables entering or leaving the basis, and the speedups are generally minimal after two or three cores [25]. For interior point methods, commercial solvers can use multiple threads to factor the associated linear systems, and speedups are generally negligible after at most six shared-memory cores, although further scaling is occasionally possible.

Another classic approach to solving large-scale LPs is to use decomposition algorithms, such as Dantzig-Wolfe decomposition, Benders decomposition, etc. While these algorithms are useful for solving some problems, they are tied to certain block structures of the problem instances and suffer from slow tail convergence, and thus they are not suitable for general-purpose LP solvers.

Given the above limitations of existing methods, first-order methods (FOMs) have become increasingly attractive for large LPs. FOMs only utilize gradient information to update their iterates, and the computational bottleneck is matrix-vector multiplication, as opposed to matrix factorization. Therefore, one only needs to store the LP instance in memory when using FOMs rather than storing any factorization. Furthermore, thanks in part to recent developments in machine learning/deep learning, FOMs scale very well on GPUs and distributed computing platforms.

Using FOMs for LP is not a new idea. As early as the 1950s, initial efforts were made to employ FOMs for solving LP problems. Notably, with the intuition to make big jumps rather than “crawling along edges”, Brown and Koopmans [9] discussed steepest ascent to maximize the linear objective under linear inequality constraints. Zoutendijk [56, 57] pioneered the development of feasible direction methods for LP, where the iterates consistently move along a feasible descent direction. The subsequent development of steepest descent gravitational methods in [12] also falls within this category. Another early approach to first-order methods for solving LP is the utilization of projected gradient algorithms [48, 28]. All these methods, however, still require solving linear systems to determine an appropriate direction for advancement. Solving the linear systems that arise during the update can be highly challenging for large instances. Moreover, the computation of projections onto polyhedral constraints can be arduous and even intractable for large-scale instances.

The recent surge of interest in large-scale applications of LP has prompted the development of new FOM-based LP algorithms, aiming to further scale up/speed up LP. The four main solvers are:

PDLP [2]. PDLP utilizes a primal-dual hybrid gradient (PDHG) method as its base algorithm and introduces practical algorithmic enhancements, such as presolving, preconditioning, adaptive restart, adaptive choice of step size, and primal weight, on top of PDHG. Right now, it has

three implementations: a prototype implemented in Julia ([FirstOrderLp.jl](#)) for research purposes, a production-level C++ implementation that is included in Google [OR-Tools](#), and an internally distributed version at Google. The internally distributed version of PDLP has been used to solve real-world problems with as many as 92B non-zeros [34], which is one of the largest LP instances ever to be solved by a general-purpose LP solver.

ABIP [29, 20]. ABIP is an alternating direction method of multipliers (ADMM)-based IPM. The core algorithm of ABIP is an homogeneous self-dual embedded interior-point method. Instead of approximately minimizing the log-barrier penalty function with a Newton’s step, ABIP utilizes multiple steps of ADMM. The $\mathcal{O}(\frac{1}{\epsilon} \log(\frac{1}{\epsilon}))$ sublinear complexity of ABIP was presented in [29]. Recently, Deng et al. [20] includes new enhancements, i.e., preconditioning, restart and hybrid parameter tuning on top of ABIP; the enhanced version is called ABIP+. ABIP+ is numerically comparable to the Julia implementation of PDLP. ABIP+ now also supports a more general conic setting when the proximal problem associated with the log-barrier in ABIP can be efficiently computed.

ECLIPSE [6]. ECLIPSE is a distributed LP solver designed specifically for large-scale LPs encountered in web applications. These LPs have a certain decomposition structure, and the effective constraints are usually much less than the number of variables. ECLIPSE looks at a certain dual formulation of the problem, then utilizes accelerated gradient descent to solve the dual problem, smoothed via Nesterov smoothing. This approach is shown to have $\mathcal{O}(\frac{1}{\epsilon})$ complexity, and it is used to solve web applications with 10^{12} decision variables [6] and real-world web applications on the LinkedIn platform [45, 1].

SCS [41, 40]. The splitting conic solver (SCS) tackles the homogeneous self-dual embedding of general conic programming using ADMM. As a special case of conic programming, SCS can also be used to solve LP. Each iteration of SCS involves projecting onto the cone and solving a system of linear equations with similar forms so that it only needs to store one factorization in memory. Furthermore, SCS supports solving the linear equations with an iterative method, which only uses matrix-vector multiplication.

In the rest of this article, we focus on discussing the theoretical and computational results of PDLP to provide a solid introduction to the field of FOMs for LP. Indeed, many theoretical guarantees we mention below can be applied directly to SCS, since ADMM is a pre-conditioned version of PDHG. Furthermore, many enhancements proposed in PDLP, such as preconditioning and restart, have been implemented in ABIP+.

2 PDHG for LP

For this section, we look at the standard form of LP for the sake of simplicity, and most of these results can be extended

to other forms of LP. More formally, we consider

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \end{aligned} \quad (1)$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. The dual of eq. (1) is given by

$$\begin{aligned} \max_{y \in \mathbb{R}^m} \quad & b^T y \\ \text{s.t.} \quad & A^T y \leq c. \end{aligned} \quad (2)$$

We start by discussing the most natural FOMs to solve LP as well as their limitations, which then lead to PDHG. Next, we discuss the convergence guarantees of vanilla PDHG for LP, and building upon that, we present the restarted PDHG, which is provably an optimal FOM for solving LP, i.e., it matches the complexity lower bound. Then, we discuss how PDHG can detect infeasibility without additional effort.

2.1 Initial attempts

The most natural FOM for solving a constrained optimization problem, such as LP (1), is perhaps the projected gradient descent (PGD), which consists of the iterative update

$$x^{k+1} = \text{proj}_{\{x \in \mathbb{R}_+^n \mid Ax=b\}}(x^k - \eta c),$$

where η is the step-size of the algorithm. All the nice theoretical guarantees of PGD for general constrained convex optimization problems can be directly applied to LP. Unfortunately, computing the projection onto the constrained set (i.e., the intersection of an affine subspace and the positive orthant) involves solving a quadratic programming problem, which can be as hard as solving the original LP, and thus PGD is not a practical algorithm for LP. To disentangle linear constraints and (simple) nonnegativity of variables, a natural idea is to dualize the linear constraints $Ax = b$ and consider the primal-dual form of the problem,

$$\min_{x \geq 0} \max_y L(x, y) := c^T x - y^T Ax + b^T y. \quad (3)$$

Convex duality theory [8] shows that the saddle points to (3) can recover the optimal solutions to the primal problem (1) and the dual problem (2). For the primal-dual formulation of LP (3), the most natural FOM is perhaps the projected gradient descent-ascent method (GDA), which iteratively updates

$$\begin{cases} x^{k+1} = \text{proj}_{\mathbb{R}_+^n}(x^k + \eta A^T y^k - \eta c) \\ y^{k+1} = y^k - \sigma A x^k + \sigma b, \end{cases}$$

where η and σ are the primal and the dual step-size, respectively. The projection of GDA is onto the positive orthant for the primal variables and it is cheap to implement. Unfortunately, GDA does not converge to a saddle point of (3). For instance, Figure 1 plots the trajectory of GDA on a simple primal-dual form of LP

$$\min_{x \geq 0} \max_y (x - 3)y, \quad (4)$$

where (3, 0) is the unique saddle point. As we can see, the GDA iterates diverge and spin away from the saddle point. Thus, GDA is not a good algorithm for solving (3).

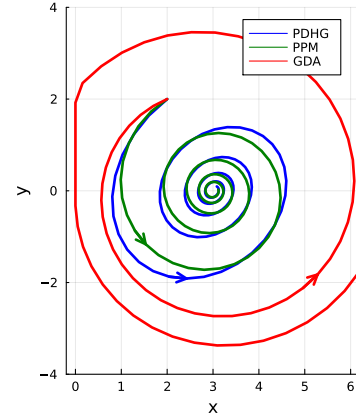


Figure 1: Trajectories of GDA, PPM and PDHG to solve a simple bilinear problem (4) with initial solution (2, 2) and step-size $\eta = \sigma = 0.2$.

Another candidate algorithm to solve (3) is the proximal point method (PPM), proposed in the seminal work of Rockafellar [47]. PPM consists of the iteration

$$(x^{k+1}, y^{k+1}) \leftarrow \arg \min_{x \geq 0} \max_y \left(L(x, y) + \frac{1}{2\eta} \|x - x^k\|_2^2 - \frac{1}{2\sigma} \|y - y^k\|_2^2 \right). \quad (5)$$

Unlike GDA, PPM exhibits nice theoretical properties for solving primal-dual problems (see Figure 1 for example). However, its update rule is implicit and requires solving the subproblems arising in (5). This drawback makes PPM more of a conceptual, rather than a practical, algorithm.

To overcome these issues of GDA and PPM, we consider the primal-dual hybrid gradient (PDHG) method, also known as the Chambolle-Pock algorithm [10, 54]. PDHG is a first-order method for convex-concave primal-dual problems originally motivated by applications in image processing. In the case of LP, the update rule is straightforward:

$$\begin{cases} x^{k+1} \leftarrow \text{proj}_{\mathbb{R}_+^n}(x^k + \eta A^T y^k - \eta c) \\ y^{k+1} \leftarrow y^k - \sigma A(2x^{k+1} - x^k) + \sigma b, \end{cases} \quad (6)$$

where η is the primal step-size and σ is the dual step-size. Similar to GDA, the algorithm alternates between the primal and the dual variables; the difference is that in the dual update, one utilizes the gradient at the extrapolated point $2x^{k+1} - x^k$. The extrapolation helps with the convergence of the algorithm, as we can see in Figure 1. Indeed, one can show PDHG is a preconditioned version of PPM (see the next section for more details), and thus enjoys the nice convergence properties of PPM. However, PDHG does not require solving the implicit update (5). The computational bottleneck of PDHG is the matrix-vector multiplication (i.e., in $A^T y$ and Ax).

2.2 Theory of PDHG on LP

In this section, we start with presenting the sublinear convergence rate for the average iterates and the last iterates of PDHG. Then we present the sharpness of the primal-dual

formulation of LP that leads to the linear convergence of PDHG on LP.

For simplicity of exposition, we assume the primal and dual step-sizes are equal, i.e., $\eta = \sigma = s$ throughout this section and the next section. This can be done without loss of generality by rescaling the primal and the dual variables. Furthermore, we assume the LP instances are feasible and bounded in this and the next section, and we will discuss infeasibility detection later on. For notational simplicity, we denote $z = (x, y)$ as the pair of the primal and dual solution,

$$P_s = \begin{pmatrix} \frac{1}{s}I & A^T \\ A & \frac{1}{s}I \end{pmatrix}, \quad \text{and} \quad \|z\|_{P_s} = \sqrt{\langle z, P_s z \rangle}.$$

The P_s norm is the inherent norm for PDHG, and it plays a critical role in the theoretical analysis of PDHG. One can clearly see this by noticing that the PDHG update (6) can be rewritten as

$$P_s(z^k - z^{k+1}) \in \mathcal{F}(z^{k+1}), \quad (7)$$

where $\mathcal{F}(z) = \begin{pmatrix} \partial_x L(x, y) \\ -\partial_y L(x, y) \end{pmatrix}$ is the sub-differential of the objective. This also showcases that PDHG is a pre-conditioned version of PPM with P_s norm by noticing that one can rewrite the update rule of PPM as $\frac{1}{s}(z^k - z^{k+1}) \in \mathcal{F}(z^{k+1})$. Armed with this understanding, one can easily obtain the sublinear convergence of PDHG.

Theorem 1 (Average iterate convergence of PDHG [10, 11, 31]). *Consider the iterates $\{z^k\}_{k=0}^\infty$ of PDHG (6) for solving (3). Denote $\bar{z}^k = (\bar{x}^k, \bar{y}^k) = \frac{1}{k} \sum_{i=1}^k z^i$ as the average iterates. Then it holds for any $0 < s \leq \frac{1}{\|A\|_2}$, $k \geq 1$, and $z = (x, y) \in \mathcal{Z}$ that*

$$L(\bar{x}^k, y) - L(x, \bar{y}^k) \leq \frac{1}{2k} \|z - z^0\|_{P_s}^2.$$

Theorem 2 (Last iterate convergence of PDHG [32]). *Consider the iterates $\{z^k\}_{k=0}^\infty$ of PDHG (6) for solving (3). Let $z^* \in \mathcal{Z}^*$ be an optimal solution to (3). Suppose the step-size satisfies $s < \frac{1}{\|A\|}$. Then, it holds for any iteration $k \geq 1$, $z = (x, y) \in \mathcal{Z}$ and any optimal solution z^* that*

$$L(x^k, y) - L(x, y^k) \leq \frac{1}{\sqrt{k}} \left(\|z^0 - z^*\|_{P_s}^2 + \|z^0 - z^*\|_{P_s} \|z^* - z\|_{P_s} \right).$$

Theorem 1 and Theorem 2 show that the average iterates of PDHG have $\mathcal{O}(1/k)$ convergence rate, and the last iterates of PDHG have $\mathcal{O}(1/\sqrt{k})$ convergence rate. The above two convergence results are not limited to LP and PDHG. [32, 31] show that these results work for convex-concave primal-dual problems (i.e., for general $L(x, y)$ as long as L is convex in x and concave in y) and for more general algorithms as long as the update rule can be written as an instance of (7), such as PPM, alternating direction method of multipliers (ADMM), etc.

Theorem 1 and Theorem 2 also imply that the average iterates of PDHG have a faster convergence rate than the last

iterates. Numerically, one often observes that the last iterates of PDHG exhibit faster convergence (even linear convergence) than the average iterates. This is due to the structure of LP, which satisfies a certain regularity condition that we call the sharpness condition. To formally define this condition, we first introduce a new progress metric, the normalized duality gap, as defined in [4].

Definition 1 (Normalized duality gap [4]). *For a primal-dual problem (3) and a solution $z = (x, y) \in \mathcal{Z}$, the normalized duality gap with radius r is defined as*

$$\rho_r(z) = \max_{\hat{z} \in W_r(z)} \frac{L(x, \hat{y}) - L(\hat{x}, y)}{r}, \quad (8)$$

where $W_r(z) = \{\hat{z} \in \mathcal{Z} \mid \|z - \hat{z}\|_2 \leq r\}$ is a ball centered at z with radius r intersected with $\mathcal{Z} = \mathbb{R}_+^n \times \mathbb{R}^m$.

Normalized duality gap is a valid progress measurement for LP, since $\rho_r(z)$ is a continuous function and $\rho_r(z) = 0$ if and only if z is an optimal solution to (3). One can show that LP is indeed a sharp problem: the normalized duality gap $\rho_r(z)$ of (3) is sharp in the standard sense.

Proposition 1 ([4]). *The primal-dual formulation of linear programming (3) is α -sharp on the set $\|z\|_2 \leq R$ for all $r \leq R$, i.e., there exists a constant $\alpha > 0$ and it holds for any z with $\|z\|_2 \leq R$ and any $r \leq R$ that*

$$\text{adist}(z, \mathcal{Z}^*) \leq \rho_r(z),$$

where \mathcal{Z}^* is the optimal solution set, $\text{dist}(z, \mathcal{Z}^*) = \min_{z^* \in \mathcal{Z}^*} \|z - z^*\|$ is the distance between z and \mathcal{Z}^* .

The next theorem shows that the last iterates of PDHG exhibit global linear convergence on LP (more generally, on sharp primal-dual problems). In contrast, the average iterates always exhibit sublinear convergence.

Theorem 3 ([32]). *Consider the iterates $\{z^k\}_{k=0}^\infty$ of PDHG (6) to solve (3). Suppose the step-size $s \leq \frac{1}{2\|A\|}$, and (3) is α -sharp on $B_R(0)$, where R is the upper bound of $\|z^k\|_2$. Then, it holds for any iteration $k \geq \lceil 4e/(s\alpha)^2 \rceil$ that*

$$\text{dist}_{P_s}(z^k, \mathcal{Z}^*) \leq \exp\left(\frac{1}{2} - \frac{k}{2 \lceil 4e/(s\alpha)^2 \rceil}\right) \text{dist}_{P_s}(z^0, \mathcal{Z}^*).$$

2.3 Optimal FOM for LP

Theorem 3 shows that the last iterates of PDHG have linear convergence with complexity $\mathcal{O}\left(\left(\frac{\|A\|_2}{\alpha}\right)^2 \log\left(\frac{1}{\epsilon}\right)\right)$ when the optimal step-size is chosen. A natural question is whether there exists FOM with faster convergence for LP. The answer is yes, and it turns out a simple variant of PDHG achieves faster (linear) convergence than PDHG matching the complexity lower bound.

Algorithm 1 formally presents this algorithm, which we dub restarted PDHG. Algorithm 1 is a two-loop algorithm. The inner loop runs PDHG until one of the restart conditions holds. At the end of each inner loop, the algorithm restarts the next outer loop from the running average of the current epoch.

Algorithm 1: Restarted PDHG for (3)

Input: Initial point (x^0, y^0) , step-sizes $0 < s < \frac{1}{\|A\|_2}$,
outer loop counter $n \leftarrow 0$.

1 repeat

2 initialize the inner loop counter $k \leftarrow 0$;

3 **repeat**

4 $x^{n,k+1} \leftarrow \text{proj}_{\mathbb{R}^n_+}(x^{n,k} + \eta A^T y^{n,k} - \eta c)$;

5 $y^{n,k+1} \leftarrow y^{n,k} - \sigma A(2x^{n,k+1} - x^{n,k}) + \sigma b$;

6 $(\bar{x}^{n,k+1}, \bar{y}^{n,k+1}) \leftarrow \frac{1}{k+1} \sum_{i=1}^{k+1} (x^{n,i}, y^{n,i})$;

7 **until** restart condition holds;

8 initialize the initial solution
 $(x^{n+1,0}, y^{n+1,0}) \leftarrow (\bar{x}^{n,k+1}, \bar{y}^{n,k+1})$;

9 $n \leftarrow n + 1$;

10 until $(x^{n+1,0}, y^{n+1,0})$ convergence;

A crucial component of the algorithm is when to restart. Suppose we know the sharpness constant α and $\|A\|_2$. The fixed frequency restart scheme proposed in [4] is to restart the algorithm every

$$k^* = \left\lceil \frac{4e\|A\|_2}{\alpha} \right\rceil \quad (9)$$

iterations. The next theorem presents the linear convergence rate of PDHG with this particular fixed frequency restart.

Theorem 4 ([4]). *Consider the iterates $\{z^{n,0}\}_{n=0}^\infty$ generated by Algorithm 1 for solving (3) with fixed frequency restart; that is, we restart the outer loop when k exceeds k^* in (9). Then for any $\epsilon > 0$, the algorithm finds $z^{n,0}$ such that $\text{dist}_{P_s}(z^{n,0}, \mathcal{Z}^*) \leq \epsilon$ within*

$$\mathcal{O}\left(\frac{\|A\|_2}{\alpha} \log\left(\frac{1}{\epsilon}\right)\right)$$

PDHG iterations.

Furthermore, [4] shows that restarted PDHG matches the complexity lower bound of a wide range of FOMs for LP. In particular, we consider *span-respecting FOMs*.

Definition 2. *An algorithm is span-respecting for an unconstrained primal-dual problem $\min_x \max_y L(x, y)$ if*

$$\begin{aligned} x^k &\in x^0 + \text{span}\{\nabla_x L(x^i, y^j) : \forall i, j \in \{1, \dots, k-1\}\} \\ y^k &\in y^0 + \text{span}\{\nabla_y L(x^i, y^j) : \forall i \in \{1, \dots, k\}, \\ &\quad \forall j \in \{1, \dots, k-1\}\}. \end{aligned}$$

Definition 2 is an extension of the span-respecting FOMs for minimization [38] in the primal-dual setting. Theorem 5 provides a lower complexity bound of span-respecting primal-dual algorithms for LP.

Theorem 5 (Lower complexity bound [4]). *Consider any iteration $k \geq 0$ and parameter value $\gamma > \alpha > 0$. There exists an α -sharp instance of LP with $\|A\|_2 = \gamma$ such that the iterates z^k of any span-respecting algorithm satisfies*

$$\text{dist}(z^k, \mathcal{Z}^*) \geq \left(1 - \frac{\alpha}{\gamma}\right)^k \text{dist}(z^0, \mathcal{Z}^*).$$

Together with Theorem 4, Theorem 5 shows that restarted PDHG is an optimal FOM for LP.

In practice, one generally does not know the sharpness constant α or the smoothness constant γ . Applegate et al. [4] proposes an adaptive restart scheme, which essentially restarts the algorithm whenever the normalized duality gap exhibits a constant factor shrinkage. The adaptive restart scheme does not require knowing the parameters of the problem, and it leads to a nearly optimal (up to a log term) complexity.

2.4 Infeasibility detection

The convergence results of PDHG in the previous section require the LP to be feasible and bounded. In practice, it is occasionally the case that an LP is infeasible or unbounded, thus infeasibility detection is a necessary feature for any LP solver. In this section, we investigate the behavior of PDHG on infeasible/unbounded LPs, and claim that the PDHG iterates encode infeasibility information automatically.

The easiest way to describe the infeasibility detection property of PDHG is perhaps to look at it from an operator perspective. More formally, we use T to represent the operator for one step of the PDHG iteration, i.e., $z^{k+1} = T(z^k)$ where T is specified by (6). Next, we introduce the infimal displacement vector of the operator T , which plays a central role in the infeasibility detection of PDHG.

Definition 3. *For the operator T induced by PDHG on (3), we call*

$$v := \arg \min_{z \in \text{range}(T-I)} \|z\|_2^2$$

its infimal displacement vector (which is uniquely defined [42]).

It turns out that if LP is primal (or dual) infeasible, then the dual (or primal) variables diverge along a ray with direction v . Furthermore, the corresponding dual (or primal) part of v provides an infeasibility certificate for the primal. Table 1 summaries such results. More formally,

Theorem 6 (Behaviors of PDHG for infeasible LP [3]). *Consider the primal problem (1) and dual problem (2). Assume $s < \frac{1}{\|A\|}$, let T be the operator induced by PDHG on (3), and let $\{z^k = (x^k, y^k)\}_{k=0}^\infty$ be a sequence generated by the fixed-point iteration from an arbitrary starting point z^0 . Then, one of the following holds:*

(a). *If both primal and dual are feasible, then the iterates (x^k, y^k) converge to a primal-dual solution $z^* = (x^*, y^*)$ and $v = (T - I)(z^*) = 0$.*

(b). *If both primal and dual are infeasible, then both primal and dual iterates diverge to infinity. Moreover, the primal and dual components of the infimal displacement vector $v = (v_x, v_y)$ give certificates of dual and primal infeasibility, respectively.*

(c). *If the primal is infeasible and the dual is feasible, then the dual iterates diverge to infinity, while the primal iterates converge to a vector x^* . The dual-component v_y is a certificate of primal infeasibility. Furthermore, there exists a vector y^* such that $v = (T - I)(x^*, y^*)$.*

(d). If the primal is feasible and the dual is infeasible, then the same conclusions as in the previous item hold by swapping primal with dual.

Furthermore, one can show that the difference of iterates and the normalized iterates converge to the infimal displacement vector v with sublinear rate:

Theorem 7 ([3, 18]). *Let T be the operator induced by PDHG on (3). Then there exists a finite z^* such that $T(z^*) = z^* + v$ and for any such z^* and all k :*

(a) (Difference of iterates)

$$\min_{j \leq k} \|v - (z^{j+1} - z^j)\| \leq \frac{1}{\sqrt{k}} \|z^0 - z^*\|_{P_s},$$

(b) (Normalized iterates)

$$\left\| v - \frac{1}{k} (z^k - z^0) \right\|_{P_s} \leq \frac{2}{k} \|z^0 - z^*\|_{P_s}.$$

Theorem 6 and Theorem 7 show that the difference of iterates and the normalized iterates of PDHG can recover the infeasibility certificates with sublinear rate. While the normalized iterates have faster sub-linear convergence than the difference of iterates, one can show that the difference of iterates converge linearly to the infimal displacement vector under additional regularity conditions [3]. In practice, PDLP periodically checks whether the difference of iterates or the normalized iterates provide an infeasibility certificate, and the performance of these two sequences is instance-dependent.

3 PDLP

In the previous section, we presented theoretical results of PDHG for LP. In the solver PDLP, there are additional algorithmic enhancements on top of PDHG to boost the practical performance. In this section, we summarize the enhancements as well as the numerical performance of the algorithms. These results were based on the Julia implementation and were presented in [2]. The algorithm in the C++ implementation of PDLP is almost identical to the Julia implementation, with two minor differences: it supports two-sided constraints, and utilizes Glop presolve instead of Papilo presolve.

PDLP solves a more general form of LP,

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{s.t.} \quad & Gx \geq h \\ & Ax = b \\ & l \leq x \leq u \end{aligned} \quad (10)$$

where $G \in \mathbb{R}^{m_1 \times n}$, $A \in \mathbb{R}^{m_2 \times n}$, $c \in \mathbb{R}^n$, $h \in \mathbb{R}^{m_1}$, $b \in \mathbb{R}^{m_2}$, $l \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{\infty\})^n$. The primal-dual form of (10) is

$$\min_{x \in X} \max_{y \in Y} L(x, y) := c^T x - y^T Kx + q^T y \quad (11)$$

where $K^T = (G^T, A^T)$, $q^T := (h^T, b^T)$, $X := \{x \in \mathbb{R}^n : l \leq x \leq u\}$, and $Y := \{y \in \mathbb{R}^{m_1+m_2} : y_{1:m_1} \geq 0\}$. In PDLP, the primal and the dual are reparameterized as

$$\eta = s/\omega, \quad \sigma = s\omega \quad \text{with } s, \omega > 0,$$

where s controls the scale of the step-size, and ω (which we call the primal weight) controls the balance between the primal and the dual variables.

3.1 Algorithmic enhancements in PDLP

PDLP has essentially five major enhancements on top of restarted PDHG: presolving, preconditioning, adaptive restart, adaptive step-size, and primal weight update.

- **Presolving.** PDLP utilizes PaPILO [22], an open-sourced library, for the presolving step. The basic idea of presolving is to simplify the problem by detecting inconsistent bounds, removing empty rows and columns of the constraint matrix, removing variables whose lower and upper bounds are equal, detecting duplicate rows and tightening bounds, etc.
- **Preconditioning.** The performance of FOMs heavily depends on the condition number. PDLP utilizes a diagonal preconditioner to improve the condition number of the problem. More specifically, PDLP rescales the constraint matrix $K = (G, A)$ to $\tilde{K} = (\tilde{G}, \tilde{A}) = D_1 K D_2$ with positive diagonal matrices D_1 and D_2 , so that the resulting matrix \tilde{K} is “well balanced”. Such preconditioning creates a new LP instance that replaces A, G, c, b, h, u and l in (10) with $\tilde{G}, \tilde{A}, \hat{x} = D_2^{-1}x, \tilde{c} = D_2c, (\tilde{b}, \tilde{h}) = D_1(b, h), \tilde{u} = D_2^{-1}u$ and $\tilde{l} = D_2^{-1}l$. In the default PDLP settings, a combination of Ruiz rescaling [49] and the preconditioning technique proposed by Pock and Chambolle [44] is applied.
- **Adaptive restarts.** PDLP utilizes an adaptive restart-scheme that is similar (but not identical) to the one we now describe. Essentially, we restart the algorithm whenever the normalized duality gap exhibits a constant-factor shrinkage,

$$\rho_{\|\bar{z}^{n,k} - z^{n,0}\|_2}(\bar{z}^{n,k}) \leq \frac{1}{2} \rho_{\|z^{n,0} - z^{n-1,0}\|_2}(\bar{z}^{n,0}), \quad (12)$$

where we have used 1/2 for simplicity. The normalized duality gap for LP can be computed with a linear time algorithm; thus this adaptive scheme can be efficiently implemented. Adaptive restarts can speed up the convergence of PDLP in order to more efficiently find high-accuracy solutions.

- **Adaptive step-size.** The theory-suggested step-size $1/\|A\|_2$ turns out to be too conservative in practice. PDLP tries to find a step-size by a heuristic line search that satisfies

$$s \leq \frac{\|z^{k+1} - z^k\|_\omega^2}{2(y^{k+1} - y^k)^T K(x^{k+1} - x^k)}, \quad (13)$$

where $\|z\|_\omega := \sqrt{w\|x\|_2^2 + \frac{\|y\|_2^2}{w}}$ and w is the current primal weight. More details of the adaptive step-size rule can be found in [2]. The inequality (13) is inspired

	Dual	
Primal		
Feasible	x^k, y^k both converge	x^k diverges, y^k converges
Infeasible	x^k converges, y^k diverges	x^k, y^k both diverge

Table 1: Behavior of PDHG for solving (3) under different feasibility assumptions.

by the $\mathcal{O}(1/k)$ convergence rate proof of PDHG [11, 31]. Employing an adaptive step-size rule results in the loss of theoretical guarantees for PDLP, but it reliably works in our numerical experiments.

- **Primal weight update.** The primal weight ω aims to heuristically balance primal and dual progress, and it is updated infrequently, in particular only when restarts occur. A detailed description of primal weight updates can be found in [2].

3.2 Numerical performance of PDLP

To illustrate the numerical performance of PDLP, we here present two sets of computational results on LP benchmark sets using the Julia implementation; these results were presented in [2]. The experiments were performed on three datasets: 383 instances from the root-node relaxation of MIPLIB 2017 collection [23] (which we dub MIP Relaxations), 56 instances from Mittelman’s benchmark set [35] (which we dub LP Benchmark), and Netlib LP benchmark [21] (which we dub Netlib). The progress metric we use is the relative KKT error, i.e., primal feasibility, dual feasibility and primal-dual gap, in the relative sense (see [2] for a more formal definition).

The first experiment is to demonstrate the effectiveness of the enhancements over vanilla PDHG. Figure 2 presents the relative improvements compared to vanilla PDHG by sequentially adding the enhancements. The y-axes of Figure 2 display the shifted geometric mean (shifted by value 10) of the KKT passes normalized by the value for vanilla PDHG. As we can see, with the exception of presolve for LP benchmark at tolerance 10^{-4} , each of our enhancements in Section 3.1 improves the performance of PDHG.

Figure 3 compares PDLP with other first-order methods: SCS [41], in both direct mode (i.e., solving the linear equation with factorization) and matrix-free mode (i.e., solving the linear equation with conjugate gradient), and our enhanced implementation of the extragradient method [27, 37]. The extragradient method is a special case of mirror prox, and it is an approximation of PPM. Restarted extragradient has similar theoretical results as restarted PDHG [4]. The comparisons are summarized in Figure 3. We can see that PDLP has superior performance in attaining both moderate accuracy 10^{-4} and high accuracy 10^{-8} when compared with the others.

4 Case studies of large LP

In this section, we present three case studies of PDLP on large (i.e., containing more than 10 million nonzeros) LP instances and compare its performance with Gurobi: personalized marketing, page rank, and robust production inventory problem. PDLP has superior performances in the

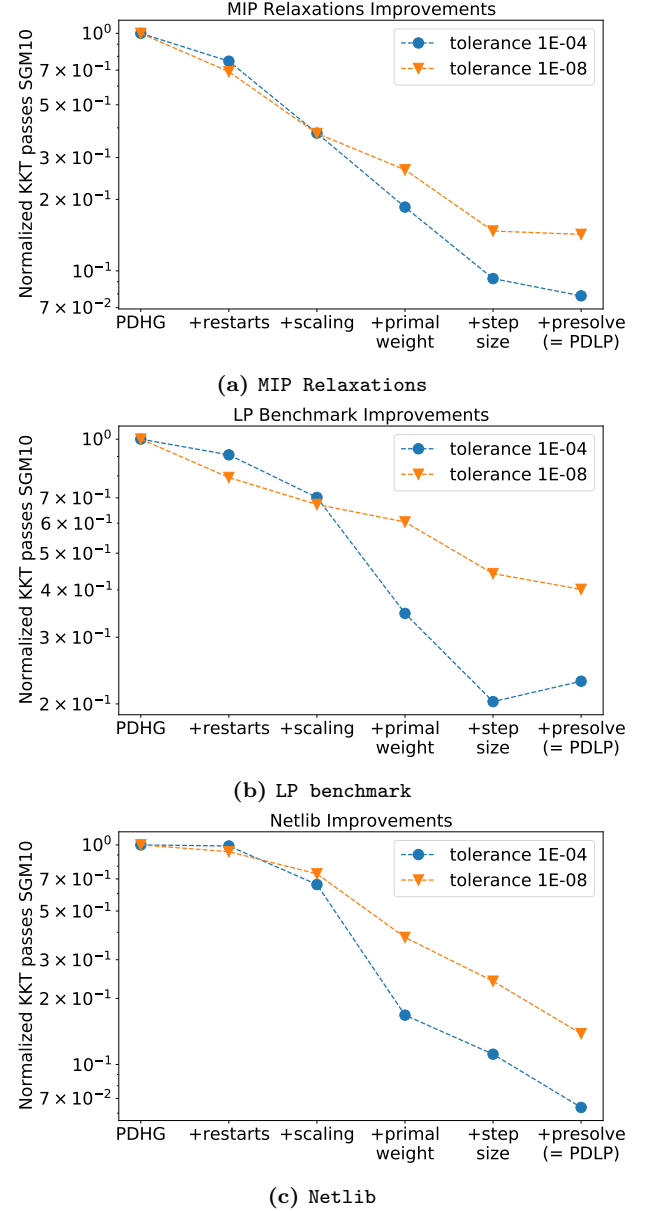


Figure 2: Summary of relative impact of PDLP’s improvements

first two instances, and Gurobi primal simplex has superior performance in the third instance. Overall, we conclude that for large instances for which factorizations can fit in memory, PDLP will not completely replace traditional LP solvers; however, it is reasonable to consider running them together in a portfolio. On the other hand, for problems for which factorization cannot fit in memory, FOMs may be the only option.

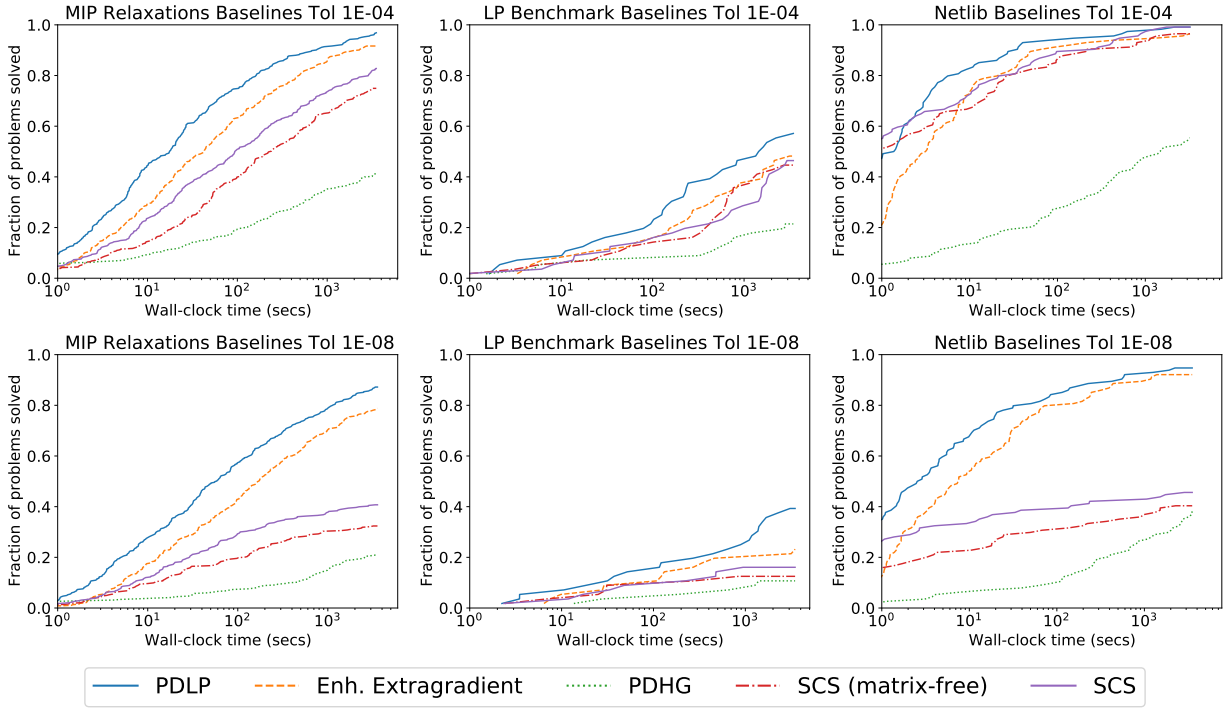


Figure 3: Number of problems solved for MIP Relaxations (left), LP benchmark (middle), and Netlib (right) datasets within one hour time limit.

4.1 Personalized marketing

Personalized marketing refers to companies sending out marketing treatments (such as discounts, coupons, etc) to individual customers periodically in order to attract more business. Such marketing treatments are usually limited by the total number of coupons that can be sent out, fairness considerations, etc. [55] proposes solving the problem using linear programming, and demonstrates the effectiveness of the model by utilizing the data of a department store collected from two states in the US.

More formally, suppose there are I target households and J available marketing actions. The decision variables, $x_i^j \in [0, 1]$, represent the probability a given household i receives marketing action j . We denote the incremental profit that the firm earns from household i if it receives marketing action j as r_i^j (these values can be suggested, for instance, by a machine learning model). The first set of constraints represents volume constraints on each marketing action, and S_k is the set of households in customer segment k . The second set of constraints captures the volume constraint on all marketing actions. The combination of the marketing actions is determined by parameter c_i^j . The third set of constraints similarities between each marketing action. The total number of households in customer segment k is denoted by n_k , and $\lambda_j^{k_1 k_2}$ restricts the difference between customer segments k_1 and k_2 . The fourth set of constraints imposes similarity constraints on all marketing actions. The difference between customer segments k_1 and k_2 is restricted by $\gamma^{k_1 k_2}$, and d_i^j is the weighting factor to determine the combination of all marketing actions. The last two constraints restrict the firm's action space so that each household has at most one marketing action. The problem can be then formulated as in

(14). Table 2 summarizes the computation time of PDLP (C++ version) versus Gurobi (primal simplex, dual simplex, and barrier method) for four different models to 10^{-4} relative accuracy. For these instances, PDLP clearly shows its advantages, and it is often the case that Gurobi runs out of memory.

4.2 PageRank

PageRank refers to the well-known problem of ranking web pages in search engine results. There are multiple ways to model the problem, one of which is to formulate the problem as finding a maximal right eigenvector of a stochastic matrix S as a feasible solution of an LP [39], that is we search for x feasible in

$$\begin{aligned} Sx &\leq x \\ \mathbf{1}^T x &= 1 \\ x &\geq 0 \end{aligned} \quad (15)$$

Nesterov [39] stated the constraint $\|x\|_\infty \geq 1$ to enforce $x \neq 0$. We instead use $\mathbf{1}^T x = 1$ since it leads to a single linear constraint.

Applegate et al. [2] generated a random scalable collection of PageRank instances, using Barabási-Albert [5] preferential attachment graphs, and the Julia `LightGraphs.SimpleGraphs.barabasi_albert` generator with degree set to 3. More specifically, an adjacency matrix is computed and scaled in the columns to make the matrix stochastic and this matrix is called S' . Following the standard PageRank formulation, a damping factor is applied to S' , $S := \lambda S' + (1 - \lambda)J/n$, where $J = \mathbf{1}\mathbf{1}^T$ is all-ones matrix. Intuitively, S encodes a random walk that follows a link in the graph with probability λ or jumps to a uniformly

$$\begin{aligned}
& \max_{x_i^j} \quad \sum_{i=1}^I \sum_{j=1}^J r_i^j x_i^j \\
& \text{s.t.} \quad a_k^j \leq \sum_{i \in S_k} x_i^j \leq b_k^j \quad \forall j \in [J], k \in [K] \\
& \quad L_k \leq \sum_{i \in S_k} \sum_{j=1}^J c_i^j x_i^j \leq U_k \quad \forall k \in [K] \\
& \quad \frac{1}{n_{k_1}} \sum_{i \in S_{k_1}} x_i^j \leq \lambda_j^{k_1 k_2} \frac{1}{n_{k_2}} \sum_{i \in S_{k_2}} x_i^j \quad \forall j \in [J], k_1 \in [K], k_2 \in [K] \\
& \quad \frac{1}{n_{k_1}} \sum_{i \in S_{k_1}} \sum_{j=1}^J d_i^j x_i^j \leq \gamma^{k_1 k_2} \frac{1}{n_{k_2}} \sum_{i \in S_{k_2}} \sum_{j=1}^J d_i^j x_i^j \quad \forall k_1 \in [K], k_2 \in [K] \\
& \quad \sum_{j=1}^J x_i^j \leq 1 \quad \forall i \in [I] \\
& \quad x_i^j \geq 0
\end{aligned} \tag{14}$$

Model	# nonzeros	Computation Time of Gurobi / PDLP (in seconds)			
		Gurobi Primal Simplex	Gurobi Dual Simplex	Gurobi Barrier	PDLP
model A	25m	15488	18469	16573	175
model B	37m	31138	-	-	341
model C	25m	-	-	-	136
model D	13m	-	-	-	127

Table 2: Solve time in second of Gurobi and PDLP for four personalized marketing instances to 10^{-4} relative accuracy. “-” refers to raising an out-of-memory error.

random node with probability $1 - \lambda$. The direct approach to the damping factor results in a completely dense matrix. However, using the fact that $Jx = 1$, we may rewrite the constraint $Sx \leq x$ in (15) as

$$\lambda(S'x)_i + (1 - \lambda)/n \leq x_i, \forall i.$$

The results are summarized in Table 3. As we can see, when the instances get 10 times larger, the running time of PDLP and SCS roughly scale linearly as the size of the instance, whereas Gurobi barrier, primal simplex, and dual simplex scale much more poorly. The fundamental reason for this is that the factorizations in the barrier and the simplex methods turn out to be much denser than the original constraint matrix. In this case, FOMs such as PDLP and SCS have clear advantages over simplex and barrier methods. We also would like to highlight that LP may not be the best way to solve PageRank problems, but PageRank provides a convenient means to generate reasonable LPs of different sizes.

4.3 Robust production inventory problem

The production inventory problem studies how to order products from factories to satisfy uncertain demand for a single product over a selling season. Ben-Tal et al. [7] proposes a linear decision rule for solving a robust version of this problem, which initiated a new trend of research in robust optimization. The robust problem can be formulated as in (16), where E is the number of factors, T is the number of

factories, ζ_t is the uncertain demand that comes from an uncertainty set \mathcal{U}_t , $y_{t,s,e}$ is the decision variable in the linear decision rule, c_{te} is the per-unit cost, Q_e is the maximum total production level of factory e , p_{te} is the maximum production level for factory e in time period t , and V_{\min} and V_{\max} specify bounds on remaining inventory level at each time period. This problem can be formulated as an LP by dualizing the inner maximization problem.

We compared the numerical performance of PDLP and Gurobi on a robust production inventory LP instance with 19 million non-zeros. While Gurobi dual simplex and barrier method failed to solve the problem within a one-day time limit, primal simplex solves it to optimality with 531160 iterations in 11800 seconds, and PDLP solves the problem to 10^{-4} accuracy with 358464 iterations in 26514 seconds. For this instance, Gurobi primal simplex outperforms PDLP. Our belief is that this problem is highly degenerate, and the optimal solution space is large. A primal-simplex method just needs to find one of the optimal extreme points, which can be done efficiently.

5 Open questions

It is an exciting time to see how FOMs can significantly scale up LP, an optimization problem that has been extensively studied since 1940s. This is indeed just the beginning of the application of FOMs to LP. There are still many open questions in this area, and we here mention three of them.

First, while [4] presents an optimal FOM for LP, which

# nodes	PDLP	SCS	Gurobi Barrier	Gurobi Primal Simp.	Gurobi Dual Simp.
10^4	7.4 sec.	1.3 sec.	36 sec.	37 sec.	114 sec.
10^5	35 sec.	38 sec.	7.8 hr.	9.3 hr.	>24 hr.
10^6	11 min.	25 min.	OOM	>24 hr.	-
10^7	5.4 hr.	3.8 hr.	-	-	-

Table 3: Solve time for PageRank instances. Gurobi barrier has crossover disabled, 1 thread. PDLP and SCS solve to 10^{-8} relative accuracy. SCS is matrix-free. Baseline PDHG is unable to solve any instances. Presolve not applied. OOM = Out of Memory. The number of nonzero coefficients per instance is $8 \times (\# \text{ nodes}) - 18$.

$$\begin{aligned}
& \min_{y_{t,1}, \dots, y_{t,t} \in \mathbb{R}^E: \forall t \in [T]} \max_{\zeta_1 \in \mathcal{U}_1, \dots, \zeta_{T+1} \in \mathcal{U}_{T+1}} \left\{ \sum_{t=1}^T \sum_{e=1}^E c_{te} \left(\sum_{s=1}^t y_{t,s,e} \zeta_s \right) \right\} \\
& \text{subject to} \quad \sum_{t=1}^T \left(\sum_{s=1}^t y_{t,s,e} \zeta_s \right) \leq Q_e \quad \forall e \in [E] \\
& \quad \quad \quad 0 \leq \left(\sum_{s=1}^t y_{t,s,e} \zeta_s \right) \leq p_{te} \quad \forall e \in [E], t \in [T] \\
& \quad \quad \quad V_{\min} \leq v_1 + \sum_{\ell=1}^t \sum_{e=1}^E \left(\sum_{s=1}^{\ell} y_{\ell,s,e} \zeta_s \right) - \sum_{s=2}^{t+1} \zeta_s \leq V_{\max} \quad \forall t \in [T] \\
& \quad \quad \quad \forall \zeta_1 \in \mathcal{U}_1, \dots, \zeta_{T+1} \in \mathcal{U}_{T+1},
\end{aligned} \tag{16}$$

matches the complexity lower bound, the linear convergence rate depends on Hoffman’s constant of the KKT system, which is known to be exponentially loose (since it considers the minimum over exponentially many items [43]), and clearly cannot characterize the numerical success of PDLP. A natural question is: can we characterize the global convergence of PDHG for LP without using Hoffman’s constant, or in other words, what is the fundamental geometric quantity of the LP instance that drives the convergence of PDHG (or more generally, FOMs)?

Second, state-of-the-art solvers for other continuous optimization problems, such as quadratic programming, second-order-cone programming, semi-definite programming and nonlinear programming, are mostly interior-point algorithms. While SCS can also be used to solve some of these problems, it still needs to solve linear equations. How much can the success of FOMs for LP be extended to other optimization problems? What are the “right” FOMs for these problems?

Third, how can FOMs be used to scale up mixed-integer programming (MIP)? In order to utilize the solution of FOMs for LP in a branch-and-bound tree, it is favorable to obtain an optimal basic feasible solution (BFS) to node LPs; FOMs typically do not directly output a BFS. Is there an efficient approach to obtain an optimal BFS from the optimal solution returned by a FOM? Furthermore, the LPs solved in a branch-and-bound tree are usually similar in nature. Can a FOM take advantage of the warm start of LP solutions therein?

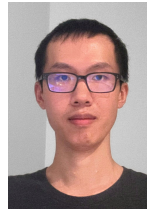
References

- [1] Ayan Acharya, Siyuan Gao, Borja Ocejó, Kinjal Basu, Ankan Saha, Keerthi Selvaraj, et al. “Promoting Inactive Members in Edge-Building Marketplace”. In: *Companion Proceedings of the ACM Web Conference 2023*. 2023, pp. 945–949.
- [2] David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O’Donoghue, et al. “Practical large-scale linear programming using primal-dual hybrid gradient”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20243–20257.
- [3] David Applegate, Mateo Díaz, Haihao Lu, and Miles Lubin. “Infeasibility detection with primal-dual hybrid gradient for large-scale linear programming”. In: *arXiv preprint arXiv:2102.04592* (2021).
- [4] David Applegate, Oliver Hinder, Haihao Lu, and Miles Lubin. “Faster first-order primal-dual methods for linear programming using restarts and sharpness”. In: *Mathematical Programming* (2022), pp. 1–52.
- [5] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *Science* 286.5439 (1999), pp. 509–512.
- [6] Kinjal Basu, Amol Ghoting, Rahul Mazumder, and Yao Pan. “Eclipse: An extreme-scale linear program solver for web-applications”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 704–714.
- [7] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitser, and Arkadi Nemirovski. “Adjustable robust solutions of uncertain linear programs”. In: *Mathematical Programming* 99.2 (2004), pp. 351–376.
- [8] Stephen Boyd and Lieven Vandenbergh. *Convex optimization*. Cambridge University Press, 2004.
- [9] George Brown and Tjalling Koopmans. “Computational suggestions for maximizing a linear function subject to linear inequalities”. In: *Activity Analysis of Production and Allocation* (1951), pp. 377–380.

- [10] Antonin Chambolle and Thomas Pock. “A first-order primal-dual algorithm for convex problems with applications to imaging”. In: *Journal of mathematical imaging and vision* 40 (2011), pp. 120–145.
- [11] Antonin Chambolle and Thomas Pock. “On the ergodic convergence rates of a first-order primal-dual algorithm”. In: *Mathematical Programming* 159.1-2 (2016), pp. 253–287.
- [12] Soo Chang and Katta Murty. “The steepest descent gravitational method for linear programming”. In: *Discrete Applied Mathematics* 25.3 (1989), pp. 211–239.
- [13] Abraham Charnes, William W Cooper, and Merton H Miller. “Application of linear programming to financial budgeting and the costing of funds”. In: *The Journal of Business* 32.1 (1959), pp. 20–46.
- [14] Munther Dahleh and Ignacio Diaz-Bobillo. *Control of uncertain systems: a linear programming approach*. Prentice-Hall, Inc., 1994.
- [15] George Dantzig. *Linear programming and extensions*. Princeton University Press, 1963.
- [16] George Dantzig. “Origins of the simplex method”. In: *A history of scientific computing*. 1990, pp. 141–151.
- [17] George B Dantzig. “Programming in a linear structure”. In: *Washington, DC* (1948).
- [18] Damek Davis and Wotao Yin. “Convergence rate analysis of several splitting schemes”. In: *Splitting methods in communication, imaging, science, and engineering* (2016), pp. 115–163.
- [19] Jerome Delson and Mohammad Shahidehpour. “Linear programming applications to power system economics, planning and operations”. In: *IEEE Transactions on Power Systems* 7.3 (1992), pp. 1155–1163.
- [20] Qi Deng, Qing Feng, Wenzhi Gao, Dongdong Ge, Bo Jiang, Yuntian Jiang, et al. “New Developments of ADMM-based Interior Point Methods for Linear Programming and Conic Programming”. In: *arXiv preprint arXiv:2209.01793* (2022).
- [21] David Gay. “Electronic mail distribution of linear programming test problems”. In: *Mathematical Programming Society COAL Newsletter* 13 (1985), pp. 10–12.
- [22] Ambros Gleixner, Leona Gottwald, and Alexander Hoen. “PaPILO: A Parallel Presolving Library for Integer and Linear Programming with Multiprecision Support”. In: *arXiv preprint arXiv:2206.10709* (2022).
- [23] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, et al. “MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library”. In: *Mathematical Programming Computation* (2021). DOI: [10.1007/s12532-020-00194-3](https://doi.org/10.1007/s12532-020-00194-3).
- [24] Peter Hazell and Pasquale Scandizzo. “Competitive demand structures under risk in agricultural linear programming models”. In: *American Journal of Agricultural Economics* 56.2 (1974), pp. 235–244.
- [25] Qi Huangfu and Julian Hall. “Parallelizing the dual revised simplex method”. In: *Mathematical Programming Computation* 10.1 (2018), pp. 119–142.
- [26] LV Kantarovich. “Mathematical methods in the organization and planning of production”. In: *Publication House of the Leningrad State University. [Translated in Management Sc. vol 66, 366-422]* (1939).
- [27] Galina M Korpelevich. “The extragradient method for finding saddle points and other problems”. In: *Matecon* 12 (1976), pp. 747–756.
- [28] Carlton Lemke. “The constrained gradient method of linear programming”. In: *Journal of the Society for Industrial and Applied Mathematics* 9.1 (1961), pp. 1–17.
- [29] Tianyi Lin, Shiqian Ma, Yinyu Ye, and Shuzhong Zhang. “An ADMM-based interior-point method for large-scale linear programming”. In: *Optimization Methods and Software* 36.2-3 (2021), pp. 389–424.
- [30] Qian Liu and Garrett Van Ryzin. “On the choice-based linear programming model for network revenue management”. In: *Manufacturing & Service Operations Management* 10.2 (2008), pp. 288–310.
- [31] Haihao Lu and Jinwen Yang. “On a Unified and Simplified Proof for the Ergodic Convergence Rates of PPM, PDHG and ADMM”. In: *arXiv preprint arXiv:2305.02165* (2023).
- [32] Haihao Lu and Jinwen Yang. “On the Infimal Sub-differential Size of Primal-Dual Hybrid Gradient Method”. In: *arXiv preprint arXiv:2206.12061* (2022).
- [33] David Luenberger and Yinyu Ye. *Linear and nonlinear programming*. Vol. 2. Springer, 1984.
- [34] Vahab Mirrokni. *Google Research, 2022 & beyond: Algorithmic advances*. <https://ai.googleblog.com/2023/02/google-research-2022-beyond-algorithmic.html>. 2023-02-10.
- [35] Hans Mittelmann. *Decision Tree for Optimization Software*. <http://plato.asu.edu/guide.html>. 2021.
- [36] Renato DC Monteiro and Ilan Adler. “Interior path following primal-dual algorithms. Part I: Linear programming”. In: *Mathematical Programming* 44.1-3 (1989), pp. 27–41.
- [37] Arkadi Nemirovski. “Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems”. In: *SIAM Journal on Optimization* 15.1 (2004), pp. 229–251.
- [38] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.
- [39] Yurii Nesterov. “Subgradient methods for huge-scale optimization problems”. In: *Mathematical Programming* 146.1-2 (2014), pp. 275–297.
- [40] Brendan O’Donoghue. “Operator splitting for a homogeneous embedding of the linear complementarity problem”. In: *SIAM Journal on Optimization* 31.3 (2021), pp. 1999–2023.

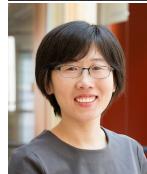
- [41] Brendan O’Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. “Conic optimization via operator splitting and homogeneous self-dual embedding”. In: *Journal of Optimization Theory and Applications* 169 (2016), pp. 1042–1068.
- [42] Amnon Pazy. “Asymptotic behavior of contractions in Hilbert space”. In: *Israel Journal of Mathematics* 9 (1971), pp. 235–240.
- [43] Javier Peña, Juan C Vera, and Luis F Zuluaga. “New characterizations of Hoffman constants for systems of linear constraints”. In: *Mathematical Programming* 187 (2021), pp. 79–109.
- [44] Thomas Pock and Antonin Chambolle. “Diagonal preconditioning for first order primal-dual algorithms in convex optimization”. In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 1762–1769.
- [45] Rohan Ramanath, S Sathiya Keerthi, Yao Pan, Konstantin Salomatin, and Kinjal Basu. “Efficient Vertex-Oriented Polytopic Projection for Web-Scale Applications”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 4. 2022, pp. 3821–3829.
- [46] James Renegar. “A polynomial-time algorithm, based on Newton’s method, for linear programming”. In: *Mathematical Programming* 40.1-3 (1988), pp. 59–93.
- [47] R Tyrrell Rockafellar. “Monotone operators and the proximal point algorithm”. In: *SIAM journal on control and optimization* 14.5 (1976), pp. 877–898.
- [48] J. Ben Rosen. “The gradient projection method for nonlinear programming. Part II. Nonlinear constraints”. In: *Journal of the Society for Industrial and Applied Mathematics* 9.4 (1961), pp. 514–532.
- [49] Daniel Ruiz. *A scaling algorithm to equilibrate both rows and columns norms in matrices*. Tech. rep. CM-P00040415, 2001.
- [50] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.
- [51] Robert Vanderbei. *Linear programming*. Springer, 2020.
- [52] Stephen Wright. *Primal-dual interior-point methods*. SIAM, 1997.
- [53] Peng Zhou and Beng Wah Ang. “Linear programming models for measuring economy-wide energy efficiency performance”. In: *Energy Policy* 36.8 (2008), pp. 2911–2916.
- [54] Mingqiang Zhu and Tony Chan. “An efficient primal-dual hybrid gradient algorithm for total variation image restoration”. In: *UCLA CAM Report* 34 (2008), pp. 8–34.
- [55] Yuting Zhu. “Augmented Machine Learning and Optimization for Marketing”. PhD thesis. Massachusetts Institute of Technology, 2022.
- [56] Guus Zoutendijk. *Methods of feasible directions*. Elsevier, Amsterdam, 1960.
- [57] Guus Zoutendijk. “Some algorithms based on the principle of feasible directions”. In: *Nonlinear programming*. Elsevier, 1970, pp. 93–121.

Global Convergence of Policy Gradient Methods in Reinforcement Learning, Games and Control



Shicong Cen

*Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh PA
shicongc@andrew.cmu.edu*



Yuejie Chi

*Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh PA
yuejiec@andrew.cmu.edu*

Policy gradient methods – methods that search for a policy of interest by maximizing value functions using first-order information – have become increasingly popular for sequential decision making in reinforcement learning, games, and control. Guaranteeing the global optimality of policy gradient methods, however, is highly nontrivial due to the general nonconcavity of value functions. In this exposition, we highlight recent progress in understanding and developing policy gradient methods with global convergence guarantees, putting an emphasis on finite-time convergence rates with regard to salient problem parameters.

1 Introduction

Sequential decision making is a canonical task that lies at the heart of a wide spectrum of disciplines such as reinforcement learning (RL), games and control. Sequential decision making finds numerous applications in autonomous driving, robotics, supply chain management, resource scheduling, and more. While classical approaches such as dynamic programming [4] requires knowledge of an underlying model, modern practice advocates for a model-free approach as an alternative. That is, one seeks a policy of interest directly based on data collected through interaction with an environment, without directly estimating the model. The advantage of the model-free approach is that it is often more memory efficient, as well as more agile to changes in the environment.

One prevalent class of model-free approaches is policy gradient (PG) methods. PG methods follow an optimizer’s perspective by formulating a value maximization problem with regard to parameterized policies. Gradient updates, often based on noisy feedback received from the environment, are employed to improve the policy iteratively. PG methods and their variants have become the de facto standard practice in an increasing number of domains, due to their seamless integration with neural network parameterization and adaptivity to various problem setups involving discrete, continuous or mixed action and state spaces.

Despite the great empirical success of PG methods, little is known about their theoretical convergence properties — especially when it comes to finite-time global convergence — due to the nonconcavity of general value functions. Not until very recently did any deep understanding begin to emerge;

recent understanding leverages the fact that PG methods are typically operated on highly structured model classes, whose induced optimization landscapes turn out to be much more benign and tractable than previously thought. The execution of problem-dependent tailored analyses, rather than relying on black-box optimization theory, fuels recent breakthroughs pioneered by Fazel et al. [14], Agarwal et al. [1], and Bhandari and Russo [5], to name just a few. The purpose of this article is to survey the latest efforts in understanding the global convergence of PG methods in the fields of RL, game theory and control, as well as highlight algorithmic ideas that enable fast global convergence, especially with regard to salient problem parameters that are crucial in practice.

Organization. The rest of this article is organized as follows. Section 2 reviews PG methods in single-agent RL, focusing on solving tabular Markov decision processes (MDP). Section 3 reviews PG methods in the game and multi-agent RL setting, using the two-player zero-sum matrix game and two-player zero-sum Markov game as illustrative examples. Section 4 moves onto PG methods in control, focusing on solving the linear quadratic regulator (LQR). We conclude in Section 5.

Notation. We use $\|A\|$, $\|A\|_F$ and $\sigma_{\min}(A)$ to represent the spectral norm, the Frobenius norm, and the smallest singular value of a matrix A , respectively. For two vectors a and b , we use $\frac{a}{b}$ to denote their entrywise division provided it exists, and $\|a\|_\infty$ denotes the entrywise maximum absolute value of a vector a . Given a set \mathcal{S} , we let $\Delta(\mathcal{S})$ represent the probability simplex over \mathcal{S} . Given two distributions p and q , $\text{KL}(p \| q)$ denotes the Kullback-Leibler (KL) divergence from q to p . Last but not least, let $\mathcal{P}_{\mathcal{C}}(\cdot)$ denote the projection operator onto the set \mathcal{C} . To characterize iteration complexity, we write $T(x) = \mathcal{O}(f(x))$ when $T(x) \leq Cf(x), \forall x \geq X$ for some $C, X > 0$. Here x is typically set to $1/\epsilon$ or some other function of salient problem parameters. We use $\tilde{\mathcal{O}}(\cdot)$ to suppress logarithmic factors from the standard order notation $\mathcal{O}(\cdot)$.

2 Global convergence of policy gradient methods in RL

In this section, we review recent progress in developing policy gradient methods for single-agent RL, focusing on the basic model of tabular Markov decision processes (MDPs).

2.1 Problem setting

Markov decision processes. An infinite-horizon discounted MDP models the sequential decision making problem as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ where we have denoted the state space \mathcal{S} , the action space \mathcal{A} , the transition kernel $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$, the reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ and the discount factor $\gamma \in (0, 1)$. Upon the agent choosing action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$, the environment will move to a new state s' according to the transition probability $P(s'|s, a)$ and assign a reward $r(s, a)$ to the agent. The action selection rule is implemented by a randomized policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, where $\pi(a|s)$ specifies the probability of choosing action a in state s . We shall focus exclusively on the case where both \mathcal{S} and \mathcal{A} are finite throughout this article.

Value functions and Q-functions. The value function $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ is defined as the expected discounted cumulative reward starting at state s :

$$V^\pi(s) := \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \right],$$

where $a_t \sim \pi(\cdot|s_t)$ is obtained by executing policy π and $s_{t+1} \sim P(\cdot|s_t, a_t)$ is generated by the MDP. The Q-function (or action-value function) $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is defined in a similar manner with an initial state-action pair (s, a) :

$$Q^\pi(s, a) := \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right].$$

In addition, the advantage function of policy π is defined as

$$A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s).$$

It is well-known that there exists an optimal policy π^* that maximizes the value function $V^\pi(s)$ for all $s \in \mathcal{S}$ simultaneously [3, 28]. We denote the resulting optimal value function and Q-function by V^* and Q^* , which satisfy the well-known Bellman optimality equations. The optimal policy π^* can be inferred from the optimal Q-function in a greedy fashion as

$$\pi^*(a|s) = \arg \max_{a \in \mathcal{A}} Q^*(s, a), \quad \forall s \in \mathcal{S}. \quad (1)$$

Policy parameterizations. Given some prescribed initial distribution ρ over \mathcal{S} , policy optimization methods seek to maximize the value function

$$V^\pi(\rho) := V^{\pi_\theta}(\rho)$$

over the policy π , where $\pi := \pi_\theta$ is often parameterized via some parameter θ . We overload the notation to denote by $V^\pi(\rho)$ the expectation of the value function $\mathbb{E}_{s \sim \rho}[V^\pi(s)]$. Note that it is straightforward to observe that $V^*(\phi) - V^\pi(\phi) \leq \|\phi/\rho\|_\infty (V^*(\rho) - V^\pi(\rho))$ for general choices of $\phi \in \Delta(\mathcal{S})$, which characterizes the effect of possible discrepancies between the initial distributions in training and deployment. We list several common choices of policy parameterization:

- *Direct parameterization:* the policy is directly parameterized by

$$\pi_\theta(a|s) = \theta(s, a),$$

where $\theta \in \{\theta \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} : \theta(s, a) \geq 0, \sum_{a \in \mathcal{A}} \theta(s, a) = 1\}$.

- *Tabular softmax parameterization:* For $\theta \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, the policy π_θ is generated through the softmax transform,

$$\pi_\theta(a|s) = \frac{\exp(\theta(s, a))}{\sum_{a' \in \mathcal{A}} \exp(\theta(s, a'))},$$

leading to an unconstrained optimization over θ . Although beyond the scope of this exposition, softmax parameterization is more popular in tandem with function approximation, since we may directly replace $\theta(s, a)$ with $f_\theta(s, a)$, the latter being typically implemented by neural networks.

Policy gradients. The gradient $\nabla_{\theta} V^{\pi_{\theta}}(\rho)$ plays an instrumental role in developing first-order policy optimization methods. To facilitate presentation, we shall first introduce the discounted state visitation distributions of a policy π ,

$$d_{s_0}^{\pi}(s) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s | s_0),$$

where the probability $\mathbb{P}(\cdot)$ is with respect to the trajectory $(s_0, a_0, s_1, a_1, \dots)$ generated by the MDP under policy π . We further denote by d_{ρ}^{π} the discounted state visitation distribution when s_0 is randomly drawn from distribution ρ , i.e.,

$$d_{\rho}^{\pi}(s) := \mathbb{E}_{s_0 \sim \rho} [d_{s_0}^{\pi}(s)].$$

The policy gradient of parameterized policy π_{θ} is then given by [35]

$$\begin{aligned} \nabla_{\theta} V^{\pi_{\theta}}(\rho) &= \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d_{\rho}^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} [\nabla_{\theta} \log \pi_{\theta}(a | s) Q^{\pi_{\theta}}(s, a)] \\ &= \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d_{\rho}^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot | s)} [\nabla_{\theta} \log \pi_{\theta}(a | s) A^{\pi_{\theta}}(s, a)], \end{aligned}$$

which can be evaluated, for example, via REINFORCE [35]. The use of the advantage function $A^{\pi_{\theta}}(s, a)$, rather than the Q-function $Q^{\pi_{\theta}}(s, a)$, in the expression for $\nabla_{\theta} V^{\pi_{\theta}}(\rho)$ often helps to reduce the variance of the estimated policy gradient.

For notational simplicity, we shall denote by $\theta^{(t)}$ and $\pi^{(t)}$ the parameter and the policy at the t -th iteration, and use $V^{(t)}$, $Q^{(t)}$, $A^{(t)}$, $d_{\rho}^{(t)}$ to denote $V^{\pi^{(t)}}$, $Q^{\pi^{(t)}}$, $A^{\pi^{(t)}}$, $d_{\rho}^{\pi^{(t)}}$, respectively. In addition, we assume the policy gradients and the value functions are deterministically evaluated throughout this article, which enables us to focus on the optimization aspect of PG methods.

2.2 Projected policy gradient method

The most straightforward first-order policy optimization method is to adopt direct parameterization and perform projected gradient ascent updates like

$$\theta^{(t+1)} = \mathcal{P}_{\Delta(\mathcal{A})|S|}(\theta^{(t)} + \eta \nabla_{\theta} V^{(t)}(\rho)), \quad (2)$$

or equivalently,

$$\pi^{(t+1)} = \mathcal{P}_{\Delta(\mathcal{A})|S|}(\pi^{(t)} + \eta \nabla_{\pi} V^{(t)}(\rho)),$$

where $\eta > 0$ is the learning rate, and

$$\nabla_{\theta(s,a)} V^{(t)}(\rho) = \nabla_{\pi(s,a)} V^{(t)}(\rho) = \frac{1}{1 - \gamma} d_{\rho}^{(t)}(s) Q^{(t)}(s, a).$$

As the value function $V^{\pi_{\theta}}(\rho)$ is $\frac{2\gamma|\mathcal{A}|}{(1-\gamma)^3}$ -smooth [1], setting the learning rate to $0 < \eta \leq \frac{(1-\gamma)^3}{2\gamma|\mathcal{A}|}$ ensures monotonicity of $V^{(t)}(\rho)$ in t . Additionally, and critically, it is established in Agarwal et al. [1] that the value function satisfies the following gradient domination condition.

Lemma 1 (Variational gradient domination). *For any policy π , we have*

$$V^{\star}(\rho) - V^{\pi}(\rho) \leq \frac{1}{1 - \gamma} \left\| \frac{d_{\rho}^{\pi^{\star}}}{\rho} \right\|_{\infty} \max_{\pi' \in \Delta(\mathcal{A})|S|} (\pi' - \pi)^{\top} \nabla_{\pi} V^{\pi}(\rho).$$

The above lemma associates the optimality gap $V^{\star}(\rho) - V^{\pi}(\rho)$ with a variational gradient term, allowing the iterates to converge globally as stated below.

Theorem 1 ([1]). *With $0 < \eta \leq \frac{(1-\gamma)^3}{2\gamma|\mathcal{A}|}$, the iterates of the projected PG method (2) satisfy*

$$\min_{0 \leq t \leq T} V^{\star}(\rho) - V^{(t)}(\rho) \leq \frac{4\sqrt{|\mathcal{S}|}}{1 - \gamma} \left\| \frac{d_{\rho}^{\pi^{\star}}}{\rho} \right\|_{\infty} \sqrt{\frac{2(V^{\star}(\rho) - V^{(0)}(\rho))}{\eta T}}.$$

Theorem 1 establishes an iteration complexity of $\mathcal{O}\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^6\epsilon^2} \left\| \frac{d_{\rho}^{\pi^{\star}}}{\rho} \right\|_{\infty}^2\right)$ for finding an ϵ -optimal policy, which was later improved to $\mathcal{O}\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^6\epsilon} \left\| \frac{d_{\rho}^{\pi^{\star}}}{\rho} \right\|_{\infty}\right)$ [36]. However, the projection operator introduces $\mathcal{O}(\log |\mathcal{A}|)$ computational overhead every iteration and is less compatible with function approximation. This motivates the study of PG methods that are compatible with unconstrained optimization, e.g., by using softmax parameterization.

2.3 Softmax policy gradient method

With softmax parameterization, the policy gradient method consists of the iteration

$$\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_{\theta} V^{(t)}(\rho), \quad (3)$$

where

$$\nabla_{\theta(s,a)} V^{(t)}(\rho) = \frac{\eta}{1 - \gamma} d_{\rho}^{(t)}(s) \pi^{(t)}(a | s) A^{(t)}(s, a).$$

Remarkably, Agarwal et al. [1] established the asymptotic global convergence of the softmax PG method (3) as follows.

Theorem 2 ([1]). *With constant learning rate $0 < \eta \leq (1 - \gamma)^3/8$, the softmax PG method (3) converges to the optimal policy, i.e., $V^{(t)}(s) \rightarrow V^{\star}(s)$ as $t \rightarrow \infty$ for all $s \in \mathcal{S}$.*

Mei et al. [24] later demonstrated an iteration complexity of $\mathcal{O}(\frac{1}{c(\mathcal{M})^2\epsilon})$ for achieving an ϵ -optimal policy, where $c(\mathcal{M})$ is a trajectory-dependent quantity depending on salient problem parameters including the number of states $|\mathcal{S}|$ and the effective horizon $(1 - \gamma)^{-1}$. Unfortunately, this quantity $c(\mathcal{M})$ can be rather small and does not exclude the possibility of incurring an excessively large iteration complexity, as demonstrated by the following hardness result [22].

Theorem 3 ([22]). *There exist universal constants $c_1, c_2, c_3 > 0$ such that for any $\gamma \in (0.96, 1)$ and $|\mathcal{S}| \geq c_3(1 - \gamma)^{-6}$, one can find a γ -discounted MDP such that the softmax PG method takes at least*

$$\frac{c_1}{\eta} |\mathcal{S}|^{2\frac{c_2}{1-\gamma}}$$

iterations to reach $\|V^{\star} - V^{(t)}\|_{\infty} \leq 0.15$.

Therefore, though guaranteed to converge globally, softmax PG can take (super-)exponential time to reduce the optimality gap to within even a constant level. To hint at the proof of this theorem, the softmax PG method fails to achieve a reasonable convergence rate when the probability $\pi^{(t)}(a^{\star}(s) | s)$

assigned to the optimal action $a^*(s)$ is close to zero. Agarwal et al. [1] proposed to penalize the policy for getting too close to the border of the simplex constraint by imposing a log barrier regularization

$$V_{\omega}^{\pi_{\theta}}(\rho) = V^{\pi_{\theta}}(\rho) + \frac{\omega}{|\mathcal{S}||\mathcal{A}|} \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \log \pi_{\theta}(a|s).$$

With an appropriate choice of regularization parameter ω , the regularized softmax PG can achieve an ϵ -optimal policy within $\mathcal{O}\left(\frac{|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^6\epsilon^2} \left\|\frac{d_{\rho}^*}{\rho}\right\|_{\infty}^2\right)$ iterations [1]. Nonetheless, this regularization scheme is not as popular in practice, compared to the entropy regularization scheme that will be discussed in Section 2.5.

2.4 Natural policy gradient method

Both PG and softmax PG fall short of attaining an iteration complexity that is independent of salient problem parameters, especially with respect to the size of the state space $|\mathcal{S}|$. This ambitious goal can be achieved, somewhat surprisingly, by employing the Fisher information matrix as a preconditioner. This preconditioning leads to the natural policy gradient (NPG) method [18],

$$\theta^{(t+1)} = \theta^{(t)} + \eta(\mathcal{F}_{\rho}^{\theta^{(t)}})^{\dagger} \nabla_{\theta} V^{(t)}(\rho), \quad (4)$$

where

$$\mathcal{F}_{\rho}^{\theta} := \mathbb{E}_{s \sim d_{\rho}^{\pi_{\theta}}, a \sim \pi_{\theta}(\cdot|s)} \left[(\nabla_{\theta} \log \pi_{\theta}(a|s)) (\nabla_{\theta} \log \pi_{\theta}(a|s))^{\top} \right]$$

is the Fisher information matrix, and \dagger denotes the Moore-Penrose pseudoinverse. With softmax parameterization, the NPG updates take the form

$$\theta^{(t+1)} = \theta^{(t)} + \frac{\eta}{1-\gamma} A^{(t)},$$

or equivalently,

$$\pi^{(t+1)}(a|s) \propto \pi^{(t)}(a|s) \exp\left(\frac{\eta Q^{(t)}(s, a)}{1-\gamma}\right).$$

It is noted that the (softmax) NPG update rule coincides with the multiplicative weights update (MWU) method [10], and that the update rule does not depend on the initial state distribution ρ . Shani, Efroni, and Mannor [30] first established a global convergence rate of $\mathcal{O}\left(\frac{1}{(1-\gamma)^2\sqrt{T}}\right)$ using decaying learning rate $\eta_t = \mathcal{O}\left(\frac{1-\gamma}{\sqrt{t}}\right)$. This result was improved by Agarwal et al. [1] using a constant learning rate η , which we now state.

Theorem 4 ([1]). *With uniform initialization $\theta^{(0)} = 0$ and constant learning rate $\eta > 0$, the iterates of NPG satisfy*

$$V^*(\rho) - V^{(T)}(\rho) \leq \frac{1}{T} \left(\frac{\log |\mathcal{A}|}{\eta} + \frac{1}{(1-\gamma)^2} \right).$$

Encouragingly, as long as $\eta \geq \frac{(1-\gamma)^2}{\log |\mathcal{A}|}$, the iteration complexity of NPG methods becomes $\mathcal{O}\left(\frac{1}{(1-\gamma)^2 T}\right)$, which is independent of the size of the state-action space. On the other hand, the iteration complexity of NPG is lower bounded

by $\frac{\Delta}{(1-\gamma)|\mathcal{A}|} \exp(-\eta\Delta T)$ — established in Khodadadian et al. [19] — where the optimal advantage function gap $\Delta = \min_s \min_{a \neq a^*(s)} |A^*(s, a)| \geq 0$ is determined by the MDP instance. As the lower bound attains its maximum $\frac{1}{(1-\gamma)e\eta T}$ when $\Delta = \frac{1}{\eta T}$, we immediately conclude that the sublinear rate in Theorem 4 cannot be improved in T . Nonetheless, two strategies to achieve even faster linear convergence with NPG updates include (i) adopting increasing/adaptive learning rates [19, 6, 20, 36], or (ii) introducing entropy regularization [7, 20, 38], which we shall elaborate on now.

2.5 Entropy regularization

Introducing entropy regularization is a popular technique in practice to promote exploration [16]. Specifically, one maximizes the entropy-regularized value function defined as

$$V_{\tau}^{\pi}(s) = V^{\pi}(s) + \frac{\tau}{1-\gamma} \mathbb{E}_{s' \sim d_s^{\pi}} [\mathcal{H}(\pi(\cdot|s'))],$$

where $\mathcal{H}(\pi(\cdot|s)) = -\sum_{a \in \mathcal{A}} \pi(a|s) \log \pi(a|s)$ is the entropy of policy $\pi(\cdot|s)$, and $\tau > 0$ serves as the regularization parameter, referred to as the temperature. The entropy-regularized Q -function is defined as

$$Q_{\tau}^{\pi}(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} [V_{\tau}^{\pi}(s')].$$

The resulting optimal value function, Q -function and optimal policy are denoted by V_{τ}^* , Q_{τ}^* , and π_{τ}^* , respectively. From an optimization perspective, the entropy term adds curvature to the value function and ensures that the optimal policy π_{τ}^* is unique. Interestingly, in contrast to the greedy optimal policy for the unregularized problem in (1), the optimal policy of the entropy-regularized problem reflects “bounded rationality” in decision making, namely

$$\pi_{\tau}^*(\cdot|s) \propto \exp(Q_{\tau}^*(s, \cdot)/\tau).$$

It should be noted, however, that adding the entropy regularization generally does not make $V_{\tau}^{\pi}(\rho)$ concave unless τ is unreasonably large. Because the entropy function is bounded by $\log |\mathcal{A}|$, the optimal entropy-regularized policy is also guaranteed to be approximately optimal for the unregularized RL problem in the following sense:

$$V^{\pi_{\tau}^*}(\rho) \geq V^*(\rho) - \frac{\tau \log |\mathcal{A}|}{1-\gamma}.$$

Motivated by its benign convergence, we consider NPG for the entropy-regularized problem. That is, we consider the iteration

$$\theta^{(t+1)} = \theta^{(t)} + \eta(\mathcal{F}_{\rho}^{\theta^{(t)}})^{\dagger} \nabla_{\theta} V_{\tau}^{(t)}(\rho),$$

which can be equivalently written as

$$\pi^{(t+1)}(a|s) \propto \pi^{(t)}(a|s)^{1-\frac{\eta\tau}{1-\gamma}} \exp\left(\frac{\eta Q_{\tau}^{(t)}(s, a)}{1-\gamma}\right). \quad (5)$$

The following theorem shows that with appropriate choices of constant learning rate η , entropy-regularized NPG converges to the unique optimal policy π_{τ}^* at a linear rate.

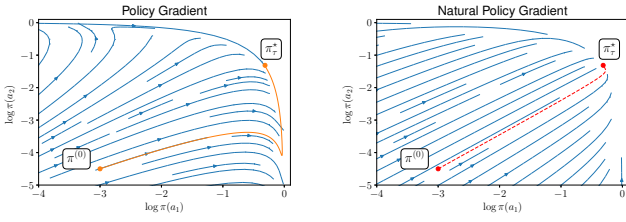


Figure 1: Comparison of PG and NPG methods with entropy regularization for a bandit problem ($\gamma = 0$) with 3 actions associated with rewards 1.0, 0.9 and 0.1. The regularization parameter is set to $\tau = 0.1$.

Theorem 5. *For constant learning rate $0 < \eta \leq (1 - \gamma)/\tau$ and uniform initialization, the entropy-regularized NPG updates (5) satisfy*

$$\|V_\tau^* - V_\tau^{(T)}\|_\infty \leq \frac{15(1 + \tau \log |\mathcal{A}|)}{1 - \gamma} (1 - \eta\tau)^{T-1}$$

and

$$V_\tau^*(\rho) - V_\tau^{(T)}(\rho) \leq \left\| \frac{\rho}{\nu_\tau^*} \right\|_\infty \left(\frac{1 + \tau \log |\mathcal{A}|}{1 - \gamma} + \frac{(1 - \gamma) \log |\mathcal{A}|}{\eta} \right) \cdot \max \left\{ \gamma, 1 - \frac{\eta\tau}{1 - \gamma} \right\}^T.$$

Here, ν_τ^* is the stationary state distribution of policy π_τ^* .

The first and the second bounds are due to Cen et al. [7] and Lan [20]¹ respectively, which lead to slightly different iteration complexities. Taken collectively, entropy-regularized NPG takes no more than

$$\tilde{O} \left(\min \left\{ \frac{1}{\eta\tau} \log \frac{1}{\epsilon}, \max \left\{ \frac{1}{1 - \gamma}, \frac{1 - \gamma}{\eta\tau} \right\} \log \frac{\|\rho/\nu_\tau^*\|_\infty}{\epsilon} \right\} \right)$$

iterations to find a policy satisfying $V_\tau^*(\rho) - V_\tau^\pi(\rho) \leq \epsilon$. We note that the difference in results stems from different analysis approaches: Cen et al. [7] built their analysis upon the contraction property of the soft Bellman operator (the entropy-regularized counterpart of the original Bellman operator), while Lan [20] made use of the connection between regularized NPG and regularized mirror descent. This latter connection to mirror descent can be understood by observing that the update rule (5) can be equivalently expressed as

$$\begin{aligned} \pi^{(t+1)}(\cdot|s) = \arg \min_{p \in \Delta(\mathcal{A})} & \langle p, -Q_\tau^{(t)}(s, \cdot) \rangle - \tau \mathcal{H}(p) \\ & + \frac{1}{\eta_{\text{MD}}} \text{KL}(p \| \pi^{(t)}(\cdot|s)), \end{aligned}$$

with $\eta_{\text{MD}} = \frac{\eta}{1 - \gamma - \eta\tau}$. The analysis of regularized RL can be further generalized to adopt non-strongly convex regularizers [20], non-smooth regularizers [38] and state-wise policy updates [21].

3 Global convergence of policy gradient methods in games

In this section, we review recent progress in developing policy gradient methods for games and multi-agent RL, focusing on

¹We discard some of the simplification steps therein and state the convergence result for a wider range of learning rate η .

the basic models of two-player zero-sum matrix games and two-player zero-sum Markov games.

3.1 Problem settings

Two-player zero-sum matrix games. Given two players, taking actions from their respective action spaces \mathcal{A} and \mathcal{B} , the zero-sum matrix game amounts to solving the saddle-point optimization problem

$$\max_{\mu \in \Delta(\mathcal{A})} \min_{\nu \in \Delta(\mathcal{B})} V^{\mu, \nu} := \mu^\top A \nu, \quad (6)$$

where $A \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{B}|}$ denotes the payoff matrix with $\|A\|_\infty \leq 1$, and $\mu \in \Delta(\mathcal{A})$ and $\nu \in \Delta(\mathcal{B})$ denote the mixed/randomized policies of each player, defined respectively as distributions over the probability simplex $\Delta(\mathcal{A})$ and $\Delta(\mathcal{B})$. Here, one player (i.e., the max player) seeks to maximize the value function while the other player (i.e., the min player) seeks to minimize it. It is well-known, dating back to Neumann [26], that the max and min operators in (6) can be exchanged without affecting the solution. A pair of policies (μ^*, ν^*) is said to be a *Nash equilibrium (NE)* of the matrix game if

$$V^{\mu^*, \nu} \geq V^{\mu^*, \nu^*} \geq V^{\mu, \nu^*} \quad \forall (\mu, \nu) \in \Delta(\mathcal{A}) \times \Delta(\mathcal{B}). \quad (7)$$

In words, the NE corresponds to when both players play their best-response strategies against their opponent.

Two-player zero-sum Markov games. Moving onto sequential decision-making, we consider an infinite-horizon discounted Markov game which is defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{B}, P, r, \gamma\}$, with discrete state space \mathcal{S} , action spaces of two players \mathcal{A} and \mathcal{B} , transition probability P , reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \rightarrow [0, 1]$ and discount factor $\gamma \in [0, 1]$. A policy $\mu : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ (resp. $\nu : \mathcal{S} \rightarrow \Delta(\mathcal{B})$) defines how the max player (resp. the min player) reacts to a given state s , where the probability of taking action $a \in \mathcal{A}$ (resp. $b \in \mathcal{B}$) is $\mu(a|s)$ (resp. $\nu(b|s)$). The transition probability kernel $P : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \rightarrow \Delta(\mathcal{S})$ defines the dynamics of the Markov game, where $P(s'|s, a, b)$ specifies the probability of transitioning to state s' from state s when the players take actions a and b respectively. The value function and Q -function of a given policy pair (μ, ν) are defined similarly as in single-agent RL, that is

$$\begin{aligned} V^{\mu, \nu}(s) &= \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, b_t) \mid s_0 = s \right], \\ Q^{\mu, \nu}(s, a, b) &= \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, b_t) \mid s_0 = s, a_0 = a, b_0 = b \right]. \end{aligned}$$

The minimax game value on state s is defined by

$$V^*(s) = \max_{\mu} \min_{\nu} V^{\mu, \nu}(s) = \min_{\nu} \max_{\mu} V^{\mu, \nu}(s).$$

Similarly, the minimax Q -function $Q^*(s, a, b)$ is defined by

$$Q^*(s, a, b) = r(s, a, b) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a, b)} V^*(s'). \quad (8)$$

It was established by [31] that there exists a pair of stationary policies (μ^*, ν^*) attaining the minimax value on all states

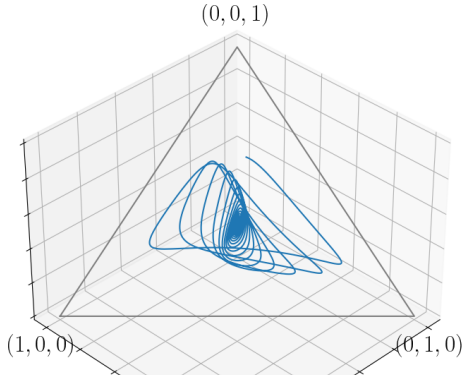


Figure 2: Cycles of MWU trajectories in a rock-paper-scissors game from a benign initialization close to the equilibrium.

[15]. This pair is called the NE of the Markov game. We seek an ϵ -optimal NE policy pair — denoted by ϵ -NE — $(\hat{\mu}^*, \hat{\nu}^*)$ that satisfies

$$V^{\mu, \hat{\nu}^*}(s) - \epsilon \leq V^{\hat{\mu}^*, \hat{\nu}^*}(s) \leq V^{\hat{\mu}^*, \mu}(s) + \epsilon$$

for any $\mu \in \Delta(\mathcal{A})^{\mathcal{S}}$, $\nu \in \Delta(\mathcal{B})^{\mathcal{S}}$ and $s \in \mathcal{S}$.

With the success of the NPG method in single-agent RL, it is tempting to apply the NPG method to two-player zero-sum games, where each player executes the NPG updates independently by treating the other player as part of their environment. Notably, the NPG dynamics coincide with the well-studied MWU method, or Hedge, that stems from online optimization and game theory, whose *average-iterate* policy (i.e., $(\frac{1}{T} \sum_{t=1}^T \mu^{(t)}, \frac{1}{T} \sum_{t=1}^T \nu^{(t)})$) is shown to achieve a convergent NE-gap at the rate of $\mathcal{O}(1/\sqrt{T})$ for two-player zero-sum matrix games. However, two technical challenges remain: 1) a demonstration of last-iterate (as opposed to average-iterate) convergence results and 2) a generalization of matrix game results to Markov games. We now discuss these two challenges separately.

3.2 Global last-iterate convergence

Average-iterate convergence guarantees fall short of determining whether the learning trajectory converges towards an NE or if the trajectory enters recurrent cycles instead. In addition, for large-scale applications that involve the use of neural networks, average-iterate convergence is also unsatisfactory since averaging neural networks is intractable. Mertikopoulos, Papadimitriou, and Piliouras [25] demonstrated that MWU, when adopted by both agents in a two-player zero-sum matrix game, suffers from the Poincaré recurrence phenomenon that prevents the method from converging. This holds for other methods in the family of “Follow the Regularized Leader” (FTRL), which necessitates algorithmic modifications to the original NPG/MWU updates.

Optimism. [29] proposed the use of *optimistic updates*, which extrapolate the gradient from the previous iteration. More specifically, in the context of two-player zero-sum matrix games, Optimistic MWU (OMWU) updates at the t -th iteration can be written as

$$\begin{cases} \mu^{(t+1)}(a) & \propto \mu^{(t)}(a) \exp(\eta(2A\nu^{(t)} - A\nu^{(t-1)})) \\ \nu^{(t+1)}(b) & \propto \nu^{(t)}(b) \exp(-\eta(2A^\top \mu^{(t)} - A^\top \mu^{(t-1)})) \end{cases}$$

Compared with the original MWU method

$$\begin{cases} \mu^{(t+1)}(a) & \propto \mu^{(t)}(a) \exp(\eta A\nu^{(t)}) \\ \nu^{(t+1)}(b) & \propto \nu^{(t)}(b) \exp(-\eta A^\top \mu^{(t)}) \end{cases},$$

OMWU estimates the reward for the next iteration by appending the terms $A\nu^{(t)} - A\nu^{(t-1)}$ and $A^\top \mu^{(t)} - A^\top \mu^{(t-1)}$ to the current payoff vectors $A\nu^{(t)}$ and $A^\top \mu^{(t)}$. The updates can be equivalently formalized as Algorithm 2, where $\{\bar{\mu}^{(t)}, \bar{\nu}^{(t)}\}_{t=0}^\infty$ are auxiliary variables that can be viewed as some “predictive” sequence for facilitating the analysis. Rakhlin and Sridharan [29] first proved that OMWU

Algorithm 2: OMWU for two-player zero-sum matrix games

- 1 **Input:** Learning rate η , (optional) regularization parameter τ .
 - 2 **Initialization:** Set $\mu^{(0)}, \nu^{(0)}, \bar{\mu}^{(0)}$ and $\bar{\nu}^{(0)}$ as uniform policies. Set $\tau = 0$ when not using regularization.
 - 3 **for** $t = 0, \dots, \infty$ **do**
 - 4 When $t \geq 1$, update $\bar{\mu}^{(t)}$ and $\bar{\nu}^{(t)}$ as

$$\begin{cases} \bar{\mu}^{(t)}(a) & \propto \bar{\mu}^{(t-1)}(a)^{1-\eta\tau} \exp(\eta A\nu^{(t)}) \\ \bar{\nu}^{(t)}(b) & \propto \bar{\nu}^{(t-1)}(b)^{1-\eta\tau} \exp(-\eta A^\top \mu^{(t)}) \end{cases}$$

Update $\mu^{(t+1)}$ and $\nu^{(t+1)}$ as

$$\begin{cases} \mu^{(t+1)}(a) & \propto \bar{\mu}^{(t)}(a)^{1-\eta\tau} \exp(\eta A\nu^{(t)}) \\ \nu^{(t+1)}(b) & \propto \bar{\nu}^{(t)}(b)^{1-\eta\tau} \exp(-\eta A^\top \mu^{(t)}) \end{cases}.$$
 - 5 **end**
-

yields $\mathcal{O}(\log T)$ regret in two-player zero-sum matrix games, thus achieving a faster $\tilde{\mathcal{O}}(1/T)$ average-iterate convergence to an NE. Daskalakis, Fishelson, and Golowich [11] demonstrated that OMWU achieves near-optimal $\mathcal{O}(\text{poly}(\log T))$ regret for multi-player general-sum games as well. Daskalakis and Panageas [13] established asymptotic last-iterate convergence of OMWU assuming that the NE is unique. Wei et al. [34] additionally demonstrated that OMWU converges linearly to the NE under the same uniqueness assumption.

Theorem 6 ([34, informal]). *For a two-player zero-sum matrix game with unique NE $\zeta^* = (\mu^*, \nu^*)$, the last iterate of OMWU $\zeta^{(T)} = (\mu^{(T)}, \nu^{(T)})$ converges to ζ^* linearly with constant learning rate $\eta \leq 1/8$.*

Wei et al. [34] also investigated optimistic gradient descent ascent (OGDA), another variant of optimistic update rules by focusing on projected gradient updates (2), and derived global linear convergence guarantees without placing assumptions on NE uniqueness. A concrete iteration complexity, however, remains elusive as both convergence rates depend on unspecified problem-dependent parameters.

Regularization. Regularization has been proven to be instrumental in enabling faster convergence for single-agent

RL. In the context of game theory, entropy regularization is closely related to quantal response equilibrium (QRE) [23], an extension to an NE with bounded rationality. Formally speaking, when the two agents seek to maximize their own entropy-regularized payoffs

$$\max_{\mu \in \Delta(\mathcal{A})} \min_{\nu \in \Delta(\mathcal{B})} V_{\tau}^{\mu, \nu} := \mu^{\top} A \nu + \tau \mathcal{H}(\mu) - \tau \mathcal{H}(\nu),$$

the resulting equilibrium $\zeta_{\tau}^* = (\mu_{\tau}^*, \nu_{\tau}^*)$ is referred to as a QRE and satisfies

$$\begin{cases} \mu_{\tau}^*(a) & \propto \exp([A \nu_{\tau}^* / \tau]_a) \\ \nu_{\tau}^*(b) & \propto \exp([A^{\top} \mu_{\tau}^* / \tau]_b) \end{cases}, \quad \forall a \in \mathcal{A}, b \in \mathcal{B}.$$

An approximate ϵ -QRE is defined similarly to an ϵ -NE, by replacing $V^{\mu, \nu}$ in (7) with its regularized counterpart $V_{\tau}^{\mu, \nu}$. An $\epsilon/2$ -QRE is guaranteed to be an ϵ -NE by setting $\tau = \tilde{\mathcal{O}}(\epsilon)$. Cen, Wei, and Chi [9] proposed entropy-regularized OMWU (summarized in Algorithm 2) to combine the ideas of regularization and optimism, and proved the method converges to a QRE at a linear rate without any assumption on the uniqueness of an NE.

Theorem 7 ([9, informal]). *With constant learning rate $0 < \eta \leq \min\{1/(2\tau + 2), 1/4\}$, the last iterate $\zeta^{(T)} = (\mu^{(T)}, \nu^{(T)})$ generated by entropy-regularized OMWU converges to a QRE ζ_{τ}^* at a linear rate $1 - \eta\tau$.*

This result gives an iteration complexity of $\mathcal{O}((1 + 1/\tau) \log 1/\epsilon)$ for finding an ϵ -QRE in a last-iterate sense, or an iteration complexity of $\tilde{\mathcal{O}}(1/\epsilon)$ for finding an ϵ -NE by setting $\tau = \tilde{\mathcal{O}}(\epsilon)$. One might wonder if using regularization alone can also ensure last-iterate convergence, which amounts to studying the update rule

$$\begin{cases} \mu^{(t)}(a) & \propto \mu^{(t-1)}(a)^{1-\eta\tau} \exp(\eta A \nu^{(t)}) \\ \nu^{(t)}(b) & \propto \nu^{(t-1)}(b)^{1-\eta\tau} \exp(-\eta A^{\top} \mu^{(t)}) \end{cases}.$$

The answer turns out to be yes, as investigated recently in Sokota et al. [32] and Pattathil, Zhang, and Ozdaglar [27]. The same contraction rate $1 - \eta\tau$ can be obtained albeit with a more restrictive choice of learning rate $\eta = \mathcal{O}(\tau)$. This result translates to an iteration complexity of $\tilde{\mathcal{O}}(1/\epsilon^2)$ for finding an ϵ -NE, slower than the entropy-regularized OMWU by a factor of $\tilde{\mathcal{O}}(1/\epsilon)$.

3.3 Extension to Markov games

Given some prescribed initial state distribution ρ , one can solve the saddle-point optimization problem

$$\max_{\mu \in \Delta(\mathcal{A})^{|\mathcal{S}|}} \min_{\nu \in \Delta(\mathcal{B})^{|\mathcal{S}|}} V^{\mu, \nu}(\rho)$$

for two-player zero-sum Markov games. A key property for two-player zero-sum matrix games is that the value function $V^{\mu, \nu}$ is bilinear in the policy space, which unfortunately no longer holds in the Markov setting. Two strategies have been successful in establishing provable policy gradient methods for two-player zero-sum Markov games: 1) leveraging tools from saddle-point optimization theory for nonconvex-nonconcave functions or 2) exploiting recursive structures of the value function.

Conventional wisdom in nonconvex-nonconcave saddle-point optimization suggests adopting two-timescale learning rates, which enforces a much smaller learning rate on one of the players. Daskalakis, Foster, and Golowich [12] demonstrated that when the two players adopt projected gradient descent ascent (GDA) updates with two-timescale learning rates, the method achieves an average-iterate convergence to an ϵ -NE within $\tilde{\mathcal{O}}(\text{poly}(\epsilon^{-1}, |\mathcal{S}|, |\mathcal{A}|, |\mathcal{B}|))$ iterations (omitting additional instance-dependent parameters). Zeng, Doan, and Romberg [37] showed that softmax policy GDA updates with entropy regularization yield a last-iterate convergence with an improved iteration complexity.

However, two-timescale learning rates give asymmetric convergence guarantees, that is, only the slow learner is guaranteed to find an approximate NE policy, while the fast learner approximates the best response to the slow learner throughout the learning process. A natural question arises: is it possible to design *symmetric* algorithms with an improved iteration complexity?

Smooth value updates. Instead of employing policy updates using only vanilla gradient information $\nabla_{\mu} V^{\mu, \nu}(\rho)$ and $\nabla_{\nu} V^{\mu, \nu}(\rho)$, another line of work seeks to divide the updates into two parts, where the policy updates are provided by two-player zero-sum matrix game algorithms with $Q^{(t)}(s) = [Q^{(t)}(s, a, b)]_{a \in \mathcal{A}, b \in \mathcal{B}}$, where

$$Q^{(t)}(s, a, b) = r(s, a, b) + \gamma \mathbb{E}_{s' \sim P(\cdot | s, a, b)} [V^{(t-1)}(s')], \quad (9a)$$

presuming the role of payoff matrices for all $s \in \mathcal{S}$, and the value updates are given by

$$V^{(t)}(s) = (1 - \alpha_t) V^{(t-1)}(s) + \alpha_t f^{(t)}(s). \quad (9b)$$

Here, $\alpha_t > 0$ is the learning rate for the value function and $f^{(t)}(s)$ is a one-step look-ahead value estimator for state s , typically defined as

$$f^{(t)}(s) = \mu^{(t)}(s)^{\top} Q^{(t)}(s) \nu^{(t)}(s), \quad (10)$$

or

$$f^{(t)}(s) = \mu^{(t)}(s)^{\top} Q^{(t)}(s) \nu^{(t)}(s) + \tau \mathcal{H}(\mu^{(t)}(s)) - \tau \mathcal{H}(\nu^{(t)}(s)) \quad (11)$$

when we incorporate entropy regularization. These methods are akin to the prevalent actor-critic type algorithms. The algorithm procedure is summarized in Algorithm 3. Note that when setting $\alpha_t = 1$ and $f^{(t)}(s)$ to the one-step minimax game value

$$f^{(t)}(s) = \max_{\mu(s) \in \Delta(\mathcal{A})} \min_{\nu(s) \in \Delta(\mathcal{B})} \mu(s)^{\top} Q^{(t)}(s) \nu(s),$$

we recover the classical value iteration for two-player zero-sum Markov games.

Wei et al. [33] first demonstrated that Algorithm 3 with **matrix_alg** OGDA and decaying learning rate $\alpha_t = \frac{2/(1-\gamma)+1}{2/(1-\gamma)+t}$ yields both average-iterate and last-iterate convergences to an NE. Cen et al. [8] achieved an improved convergence rate by adopting entropy regularization and the OMWU method (e.g. Algorithm 2).

Algorithm 3: Actor-critic for two-player zero-sum Markov games

```

1 Input: Learning rate for  $Q$ -value function  $\{\alpha_t\}_{t=0}^\infty$ ,
   learning rate for policies  $\eta$ , policy optimization
   method for two-player zero-sum matrix game
   matrix_alg, (optional) regularization parameter  $\tau$ .
2 Initialization: Set  $Q^{(0)} = 0$  and  $\mu^{(0)}, \nu^{(0)}$  as uniform
   policies.
3 for  $t = 0, 1, \dots$  do
4   for all  $s \in \mathcal{S}$  do in parallel
5     Invoke matrix_alg with payoff matrix  $Q^{(t)}(s)$ 
       and learning rate  $\eta$  to update  $\mu^{(t+1)}(s)$ ,
        $\nu^{(t+1)}(s)$ .
6     Update  $Q^{(t+1)}(s)$  and  $V^{(t+1)}(s)$  according to
       (9) with learning rate  $\alpha_{t+1}$ .
7   end
8 end

```

Theorem 8 ([8, Theorem 1]). *Algorithm 3 with `matrix_alg` entropy-regularized OMWU, entropy-regularized value updates (11) and constant learning rates $\eta = \mathcal{O}((1-\gamma)^3/|\mathcal{S}|)$, $\alpha_t = \eta\tau$ guarantees last-iterate convergence to a QRE at a linear rate $1 - \eta\tau$.*

This theorem demonstrates an iteration complexity of $\tilde{\mathcal{O}}\left(\frac{|\mathcal{S}|}{(1-\gamma)^{4\tau}} \log 1/\epsilon\right)$ for finding an ϵ -QRE, or $\tilde{\mathcal{O}}\left(\frac{|\mathcal{S}|}{(1-\gamma)^{5\epsilon}}\right)$ for finding an ϵ -NE. It remains an open problem to achieve an iteration complexity with better dependency on $|\mathcal{S}|$ and $(1-\gamma)^{-1}$.

4 Global convergence of policy gradient methods in control

In this section, we briefly review policy gradient methods for control, focusing on a standard control problem called linear quadratic regulators (LQRs). This line of research is based primarily on the excellent work of Fazel et al. [14]. We refer interested readers to Hu et al. [17] for a recent comprehensive survey on developments of policy optimization for general control problems including, but not limited to, \mathcal{H}_∞ control and risk-sensitive control.

4.1 Problem setting

Linear quadratic regulator (LQR). Consider a discrete-time linear dynamic system

$$x^{(t+1)} = Ax^{(t)} + Bu^{(t)},$$

where $x^{(t)} \in \mathbb{R}^d$, $u^{(t)} \in \mathbb{R}^k$ are respectively the state and the input at time t and $A \in \mathbb{R}^{d \times d}$, $B \in \mathbb{R}^{d \times k}$ specify system transition matrices. The linear quadratic regulator (LQR) problem with an infinite horizon is defined as

$$\min_{u^{(t)}} \mathbb{E}_{x^{(0)} \sim \mathcal{D}} \left[\sum_{t=0}^{\infty} \left(x^{(t)\top} Q x^{(t)} + u^{(t)\top} R u^{(t)} \right) \right], \quad (12)$$

where \mathcal{D} determines the distribution of the initial state $x^{(0)}$, and where $Q \in \mathbb{R}^{d \times d}$ and $R \in \mathbb{R}^{k \times k}$ are positive definite matrices parametrizing the costs. Classical optimal control theory [2] tells us that under certain stability conditions (e.g.,

controllability), it is ensured that the optimal cost is finite and can be achieved by a linear controller

$$u^{(t)} = -K^* x^{(t)},$$

where $K^* \in \mathbb{R}^{k \times d}$ is the optimal control gain matrix. From an optimization perspective, it is therefore natural to cast the LQR problem as minimizing the cost over all linear controllers $u^{(t)} = -Kx^{(t)}$ with $K \in \mathbb{R}^{k \times d}$,

$$\min_{K \in \mathcal{K}} C(K) := \mathbb{E}_{x^{(0)} \sim \mathcal{D}} \left[x^{(0)\top} P_K x^{(0)} \right],$$

where

$$P_K = \sum_{t=0}^{\infty} ((A - BK)^\top)^t (Q + K^\top R K) (A - BK)^t.$$

The feasible set $\mathcal{K} = \{K : \|A - BK\|_2 < 1\}$ ensures P_K is well defined for all $K \in \mathcal{K}$.

4.2 Policy gradient method

The policy gradient method for the LQR problem is simply defined as

$$K^{(t+1)} = K^{(t)} - \eta \nabla_K C(K^{(t)}), \quad (13)$$

where $\eta > 0$ is the learning rate. In addition, the policy gradient $\nabla_K C(K)$ can be written as

$$\nabla_K C(K) = 2((R + B^\top P_K B)K - B^\top P_K A)\Sigma_K,$$

where $\Sigma_K = \mathbb{E}_{x^{(0)} \sim \mathcal{D}} [\sum_{t=0}^{\infty} x^{(t)} x^{(t)\top}]$ denotes the state correlation matrix.

Like the RL problem, the objective function $C(K)$ is nonconvex in general, which makes it challenging to claim a global convergence guarantee. Fortunately, the LQR problem satisfies the following gradient dominance condition [14].

Lemma 2 (Gradient dominance). *Suppose that $\lambda = \sigma_{\min}(\mathbb{E}_{x^{(0)} \sim \mathcal{D}} [x^{(0)} x^{(0)\top}]) > 0$. It holds that*

$$C(K) - C(K^*) \leq \frac{\|\Sigma_{K^*}\|_2}{\lambda^2 \sigma_{\min}(R)} \|\nabla_K C(K)\|_F^2.$$

The gradient dominance property provides hope for attaining global linear convergence of the policy gradient method to the optimal policy. To complete the puzzle, however, we also desire smoothness of the cost function $C(K)$. Such smoothness cannot be established in full generality; $C(K)$ is infinite when K moves beyond \mathcal{K} . Fortunately, however, smoothness can be established for any given sublevel set $\mathcal{K}_{\bar{\gamma}} = \{K \in \mathcal{K} : C(K) \leq \bar{\gamma}\}$, which suffices to establish the following desired convergence result.

Theorem 9 (Fazel et al. [14]). *Assume that $C(K^{(0)})$ is finite. With an appropriate constant learning rate*

$$\eta = \text{poly}\left(\frac{\lambda \sigma_{\min}(Q)}{C(K^{(0)})}, \|A\|_2^{-1}, \|B\|_2^{-1}, \|R\|_2^{-1}, \sigma_{\min}(R)\right),$$

the policy gradient method (13) converges to the optimal policy at a linear rate:

$$C(K^{(t+1)}) - C(K^*) \leq \left(1 - \frac{\lambda^2 \sigma_{\min}(R) \eta}{\|\Sigma_{K^*}\|_2}\right) (C(K^{(t)}) - C(K^*)).$$

To find an ϵ -optimal control policy $K^{(T)}$ satisfying $C(K^{(T)}) - C(K^*) \leq \epsilon$, the above theorem ensures that the policy gradient method takes no more than

$$\frac{\|\Sigma_{K^*}\|_2}{\lambda^2 \sigma_{\min}(R)\eta} \log \frac{C(K^{(0)}) - C(K^*)}{\epsilon}$$

iterations.

4.3 Natural policy gradient method

To facilitate the development of an NPG in this problem setting, we consider a linear policy with additive Gaussian noise, specified as

$$u^{(t)} \sim \pi(\cdot|x^{(t)}) = \mathcal{N}(-Kx^{(t)}, \sigma^2 I).$$

The NPG update rule [18] then reads like

$$\text{vec}(K^{(t+1)}) = \text{vec}(K^{(t)}) - \eta(\mathcal{F}_{\mathcal{D}}^{K^{(t)}})^{\dagger} \text{vec}(\nabla_K C(K^{(t)})), \quad (14)$$

where $\text{vec}(K)$ flattens $K \in \mathbb{R}^{k \times d}$ into a vector in row-major order, and the Fisher information matrix $\mathcal{F}_{\mathcal{D}}^K \in \mathbb{R}^{kd \times kd}$ is given by

$$\begin{aligned} \mathcal{F}_{\mathcal{D}}^K &= \mathbb{E} \left[\sum_{t=0}^{\infty} \text{vec}(\nabla_K \log \pi(u^{(t)}|x^{(t)})) \text{vec}(\nabla_K \log \pi(u^{(t)}|x^{(t)}))^{\top} \right] \\ &= \sigma^{-2} \mathbb{E} \left[\sum_{t=0}^{\infty} \text{diag}(x^{(t)} x^{(t)\top}, \dots, x^{(t)} x^{(t)\top}) \right] \\ &= \sigma^{-2} \text{diag}(\Sigma_K, \dots, \Sigma_K). \end{aligned}$$

Merging the dummy variance σ^2 into the learning rate η , and reshaping back into the matrix form, the NPG update rule (14) can be equivalently rewritten as

$$K^{(t+1)} = K^{(t)} - \eta \nabla_K C(K^{(t)}) \Sigma_{K^{(t)}}^{-1},$$

which modifies the update direction using the state correlation matrix. This allows for an improved progress following a single update, as demonstrated by the following lemma.

Lemma 3 (Fazel et al. [14]). *Assume that $C(K^{(t)})$ is finite and the learning rate satisfies $\eta \leq \frac{1}{\|R+B^{\top}P_{K^{(t)}}B\|_2}$. Then, the NPG update satisfies*

$$C(K^{(t+1)}) - C(K^*) \leq \left(1 - \frac{\lambda \sigma_{\min}(R)\eta}{\|\Sigma_{K^*}\|_2}\right) (C(K^{(t)}) - C(K^*)).$$

The improvement is twofold: 1) the convergence rate is improved by a factor of λ , and 2) the result allows for a larger learning rate that would not be possible under an analysis based on smoothness. By adopting $\eta = \frac{1}{\|R\|_2 + \|B\|_2^2 C(K^{(0)}) \lambda^{-1}}$, one can show that the learning rate requirement is satisfied over the whole trajectory, which leads to an iteration complexity of

$$\frac{\|\Sigma_{K^*}\|_2}{\lambda \sigma_{\min}(R)\eta} \log \frac{C(K^{(0)}) - C(K^*)}{\epsilon}$$

for finding an ϵ -optimal control policy. Last, but not least, it is possible to achieve an even-faster convergence rate by assuming access to more complex oracles, i.e., those employed

in the Gauss-Newton method. The update rule in this case is given by

$$K^{(t+1)} = K^{(t)} - \eta(R + B^{\top}P_{K^{(t)}}B)^{-1} \nabla_K C(K^{(t)}) \Sigma_{K^{(t)}}^{-1},$$

which allows a constant learning rate as large as $\eta = 1$ and an improved iteration complexity of

$$\frac{\|\Sigma_{K^*}\|_2}{\lambda} \log \frac{C(K^{(0)}) - C(K^*)}{\epsilon}.$$

5 Conclusions

Policy gradient methods continue to be at the forefront of data-driven sequential decision making, due to their simplicity and flexibility in integrating with other advances in computation from adjacent fields, such as high performance computing and deep learning. Our focus in this article was constrained to the optimization aspects of policy gradient methods, assuming access to exact gradient or policy evaluations. In reality, these information need to be estimated by samples collected via various mechanisms and are thus noisy. This leads to deep interplays between statistics and optimization. We hope this exposition provides a teaser to invite more interest from the optimization community to work in this area.

Acknowledgement S. Cen and Y. Chi are supported in part by the grants ONR N00014-19-1-2404, NSF CCF-1901199, CCF-2106778 and CNS-2148212. S. Cen is also gratefully supported by Wei Shen and Xuehong Zhang Presidential Fellowship, Boeing Scholarship, and JP Morgan Chase AI PhD Fellowship.

References

- [1] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. “On the theory of policy gradient methods: Optimality, approximation, and distribution shift”. In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 4431–4506.
- [2] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.
- [3] Richard Bellman. “On the theory of dynamic programming”. In: *Proceedings of the National Academy of Sciences of the United States of America* 38.8 (1952), p. 716.
- [4] Dimitri P Bertsekas. *Dynamic programming and optimal control (4th edition)*. Athena Scientific, 2017.
- [5] Jalaj Bhandari and Daniel Russo. “Global optimality guarantees for policy gradient methods”. In: *arXiv preprint arXiv:1906.01786* (2019).
- [6] Jalaj Bhandari and Daniel Russo. “On the linear convergence of policy gradient methods for finite mdps”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2386–2394.
- [7] Shicong Cen, Chen Cheng, Yuxin Chen, Yuting Wei, and Yuejie Chi. “Fast global convergence of natural policy gradient methods with entropy regularization”. In: *Operations Research* 70.4 (2022), pp. 2563–2578.


- [8] Shicong Cen, Yuejie Chi, Simon S Du, and Lin Xiao. “Faster last-iterate convergence of policy optimization in zero-sum Markov games”. In: *International Conference on Learning Representations*. 2023.
- [9] Shicong Cen, Yuting Wei, and Yuejie Chi. “Fast policy extragradient methods for competitive games with entropy regularization”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27952–27964.
- [10] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.
- [11] Constantinos Daskalakis, Maxwell Fishelson, and Noah Golowich. “Near-optimal no-regret learning in general games”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27604–27616.
- [12] Constantinos Daskalakis, Dylan J Foster, and Noah Golowich. “Independent Policy Gradient Methods for Competitive Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5527–5540.
- [13] Constantinos Daskalakis and Ioannis Panageas. “Last-iterate convergence: Zero-sum games and constrained min-max optimization”. In: *Innovations in Theoretical Computer Science*. 2019.
- [14] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. “Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator”. In: *International Conference on Machine Learning*. 2018, pp. 1467–1476.
- [15] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.
- [16] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. “Reinforcement Learning with Deep Energy-Based Policies”. In: *International Conference on Machine Learning*. 2017, pp. 1352–1361.
- [17] Bin Hu, Kaiqing Zhang, Na Li, Mehran Mesbahi, Maryam Fazel, and Tamer Başar. “Toward a Theoretical Foundation of Policy Optimization for Learning Control Policies”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 6 (2023), pp. 123–158.
- [18] Sham M Kakade. “A natural policy gradient”. In: *Advances in neural information processing systems*. 2002, pp. 1531–1538.
- [19] Sajad Khodadadian, Prakirt Raj Jhunjhunwala, Sushil Mahavir Varma, and Siva Theja Maguluri. “On the linear convergence of natural policy gradient algorithm”. In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 3794–3799.
- [20] Guanghui Lan. “Policy mirror descent for reinforcement learning: Linear convergence, new sampling complexity, and generalized problem classes”. In: *Mathematical programming* 198.1 (2023), pp. 1059–1106.
- [21] Guanghui Lan, Yan Li, and Tuo Zhao. “Block policy mirror descent”. In: *SIAM Journal on Optimization* 33.3 (2023), pp. 2341–2378.
- [22] Gen Li, Yuting Wei, Yuejie Chi, and Yuxin Chen. “Softmax policy gradient methods can take exponential time to converge”. In: *Mathematical Programming* (2023), pp. 1–96.
- [23] Richard D McKelvey and Thomas R Palfrey. “Quantal response equilibria for normal form games”. In: *Games and economic behavior* 10.1 (1995), pp. 6–38.
- [24] Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. “On the global convergence rates of softmax policy gradient methods”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6820–6829.
- [25] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. “Cycles in adversarial regularized learning”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2018, pp. 2703–2717.
- [26] John von Neumann. “Zur Theorie der Gesellschaftsspiele”. In: *Mathematische Annalen* 100.1 (1928), pp. 295–320.
- [27] Sarath Pattathil, Kaiqing Zhang, and Asuman Ozdaglar. “Symmetric (optimistic) natural policy gradient for multi-agent learning with parameter convergence”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 5641–5685.
- [28] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [29] Sasha Rakhlin and Karthik Sridharan. “Optimization, learning, and games with predictable sequences”. In: *Advances in Neural Information Processing Systems* 26 (2013).
- [30] Lior Shani, Yonathan Efroni, and Shie Mannor. “Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdp”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5668–5675.
- [31] Lloyd S Shapley. “Stochastic games”. In: *Proceedings of the National Academy of Sciences* 39.10 (1953), pp. 1095–1100.
- [32] Samuel Sokota, Ryan D’Orazio, J Zico Kolter, Nicolas Loizou, Marc Lanctot, Ioannis Mitliagkas, et al. “A Unified Approach to Reinforcement Learning, Quantal Response Equilibria, and Two-Player Zero-Sum Games”. In: *International Conference on Learning Representations (ICLR)*. 2023.
- [33] Chen-Yu Wei, Chung-Wei Lee, Mengxiao Zhang, and Haipeng Luo. “Last-iterate convergence of decentralized optimistic gradient descent/ascent in infinite-horizon competitive Markov games”. In: *Conference on learning theory*. PMLR. 2021, pp. 4259–4299.
- [34] Chen-Yu Wei, Chung-Wei Lee, Mengxiao Zhang, and Haipeng Luo. “Linear Last-iterate Convergence in Constrained Saddle-point Optimization”. In: *International Conference on Learning Representations (ICLR)*. 2021.

- [35] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8.3-4 (1992), pp. 229–256.
- [36] Lin Xiao. “On the convergence rates of policy gradient methods”. In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 12887–12922.
- [37] Sihan Zeng, Thinh T. Doan, and Justin Romberg. “Regularized Gradient Descent Ascent for Two-Player Zero-Sum Markov Games”. In: *Advances in Neural Information Processing Systems* 35 (2022).
- [38] Wenhao Zhan, Shicong Cen, Baihe Huang, Yuxin Chen, Jason D Lee, and Yuejie Chi. “Policy mirror descent for regularized reinforcement learning: A generalized framework with linear convergence”. In: *SIAM Journal on Optimization* 33.2 (2023), pp. 1061–1091.

Bulletin

Email items to siagoptnews@lists.mcs.anl.gov for consideration in the bulletin of forthcoming issues.

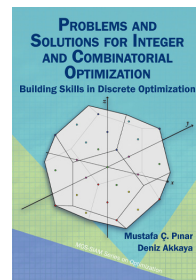
Event Announcements


**25th International Symposium
on Mathematical Programming
(ISMP)
21-26 July 2023
Montréal, Canada**

ISMP is a triennial conference that gathers scholars from around the globe to share recent developments in the field of Mathematical Optimization. The 2024 edition will take place at the Palais des congrès in Montréal, Canada. The deadlines are 29 February 2024 for stream submission, 19 April 2024 for abstract submission, and the following for registering: 29 February 2024 (early bird) and 20 June 2024 (normal).

URL: <https://ismp2024.gerad.ca>

Books



Problems and Solutions for Integer and Combinatorial Optimization: Building Skills in Discrete Optimization

by Mustafa Ç. Pinar and Deniz Akkaya

Publisher: SIAM

ISBN: 978-1-61197-775-2

Published: 2023

Series: MOS-SIAM Series on Optimization

ABOUT THE BOOK: This book contains 102 solved problems in integer and combinatorial optimization, with a wide range of difficulty and selected from a vast set of topics and applications. Among the subjects covered in the book we find modeling with integer variables, Branch and Bound, cutting planes, models for network optimization, and many others. The authors also maintain an associated website with more problems and lecture notes.

AUDIENCE: this book is meant for undergraduate and graduate students in Mathematics, Computer Science, and Engineering, and for instructors to use with other course material for Discrete Optimization courses.

Other Announcements

Gödel prize to S. Fiorini, S. Massar, S. Pokutta, H.R. Tiwary, R. de Wolf, and T. Rothvoss

The Kurt Gödel prize is awarded annually to papers in the area of theoretical computer science. It is sponsored by the European Association for Theoretical Computer Science (EATCS) and the Special Interest Group on Algorithms and Computation Theory of the ACM.

The 2023 edition of the Gödel prize has been awarded to

- Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf, for their paper “Exponential Lower Bounds for Polytopes in Combinatorial Optimization,” *STOC* 2012: 95-106 and *Journal of the ACM* 62(2), 17:1-17:23 (2015);
- Thomas Rothvoss for his paper “The matching polytope has exponential extension complexity,” *STOC* 2014: 263-272, *Journal of the ACM* 64(6), 1-19 (2017).

See <https://sigact.org/prizes/gödel.html> for more information. Congratulations to all awardees!

SIAG/OPT Test of Time Award

The SIAG/OPT Test of Time Award, established in 2022, is awarded every three years to an individual or group of researchers for an outstanding single piece of work that has had significant and sustained influence on the field of optimization over a time period of at least 10 years preceding the year of the award.

The recipients of the 2023 SIAG/OPT Test of Time Award are Samuel Burer, University of Iowa, and Renato D.C. Monteiro, Georgia Institute of Technology (Georgia Tech), for their paper, “A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization,” *Mathematical Programming Series B*, No. 95, pp. 329-357, (2003), which has revolutionized computational methods for solving certain classes of large scale semidefinite programming problems. See the [award page](#) for more information. Congratulations Sam and Renato!

SIAG/OPT Best Paper Award

The SIAG/OPT Best Paper Prize is awarded every three years to the author(s) of the most outstanding paper, as determined by the prize committee, on a topic in optimization published in the four calendar years preceding the award year.

The recipients of the 2023 SIAG/OPT Best Paper Prize are Damek Davis, Cornell University, and Dmitriy Drusvyatskiy, University of Washington for their paper, “Stochastic Model-Based Minimization of Weakly Convex Functions,” *SIAM Journal on Optimization*, Vol. 29, No. 1, pp. 207–239, (2019), which for the first time establishes convergence rates of a broad family of algorithms for the stochastic optimization of weakly convex functions. See the [award page](#) for more information. Congratulations go to Damek and to (our very own) Dmitriy!

Chair’s Column

A research community requires support and dedication, and my first words go to the past officers of our SIAG on Optimization: Katya Scheinberg (Chair), Samuel A. Burer (Vice Chair), Jeffrey Linderoth (Program Director), and Stefan M. Wild (Secretary). Thank you Katya, Sam, Jeff, and Stefan for leading us through pandemic time—it wasn’t easy for sure! The creation and successful launch of the Test of Time prize will certainly be a long lasting contribution.

We are extremely fortunate to count on three talented optimizers on our new board: Coralia Cartis (Vice Chair), Gabriele Eichfelder (Program Director), and Julianne Mueller (Secretary). It has been a great pleasure to work with Cora, Gabi, and Juli, and I hope that we can together contribute to further develop our SIAM Activity Group. I would also like to thank Nicole Gawel, Membership Coordinator at SIAM, for supporting and motivating us.

What a joy it was attending the 2023 SIAM Conference on Optimization at Seattle! A large conference like SIOPT 2023 isn’t possible without the contribution of many people, in this case from the inspirational main organizers (Coralie Cartis, Katya Scheinberg, and Jeff Linderoth) and the dedicated members of the organizing committee to the numerous session organizers and all SIAM staff members involved. To all of them, I express my sincere gratitude. The speakers and the participants did the rest, and such a rest wasn’t trivial as we had a vibrant conference, full of interesting talks, follow-up discussions, and social interaction. Here are SIOPT 2023 stats (provided by SIAM):

- Total attendance (includes ACDA 2023): 1,220 (previous peak was 694 at OP17 in Vancouver). Note that only 3% of the attendance was purely ACDA. Members of both ACDA and Optimization SIAG’s accounted for 16%. The 81% remaining were purely Optimization.
- Total number of minisymposia: 336 (x 3 talks per mini = 1,008 minisymp talks)
- Total number of contributed talks: 137
- Total number of posters: 12
- Total number of plenary/prize talks: 9
- Total number of minitutorials: 4
- Parallelism: 24 sessions in parallel

In our SIOPT 2023 business meeting, we have decided to launch a call to better identify and select the location of our 2026 SIAM Conference on Optimization. The call has been posted at SIAM Engage and at a number of other optimization forums and digests (the text is reproduced in

a postscript below). I am very much looking forward to a number of strong proposals. I take this opportunity to thank Richard O. Moore, Director of Programs and Services at SIAM, who has been very supportive during this process.

In our SIOPT 2023 business meeting, we also discussed 3 alternatives for future activities of our SIAG: (1) On-line mini-courses; (2) Regional/sectional (in-person) events; (3) On-line social hours. The idea of social hours did not attract much enthusiasm, and we have decided to not pursue it.

We are now asking for suggestions regarding on-line mini-courses. Please write to us if you have interesting ideas! Please include the topic, why the topic is timely, potential instructors, approximate duration, and potential interested audience. As we said in a recent SIAM Engage post, a questionnaire may later be sent if there is a need to sense a majority of opinions.

My final words go to the current editors of our SIAG on Optimization Views-and-News: Pietro Belotti, Dmitriy Drusvyatskiy, and Matt Menickelly. Thank you Dima, Matt, and Pietro for your outstanding work! I was the editor of our newsletter from 2003 to 2008, and I know from experience how much work is required to promote interesting articles, follow up with authors, review all materials, and edit everything together.

I wish you all a very pleasant and productive 2024, Luis

Luis Nunes Vicente (Chair, SIAG on Optimization)
Timothy J. Wilmott Endowed Chair Professor and Department Chair
Department of Industrial and Systems Engineering, Lehigh University

Comments from the Editors

Happy New Year, SIAG on Optimization! In this slightly delayed December issue, we are pleased to present two feature articles highlighting trends and developments in our field.

In our first article, Haihao Lu discusses the use of first-order methods for solving problems in linear programming (LP). While simplex methods and interior point methods are the gold standard for commercial LP solvers, the need to solve increasingly larger instances merits a reexamination of some of the practical considerations underlying these methods. In particular, many of the factorizations involved in subproblems for these standard methods involve more memory than is available to many computational platforms. Moreover, making efficient use of GPUs and massive parallelism for these standard methods is nontrivial. Contrasting these concerns with first-order methods, which require only matrix-vector multiplications (as opposed to factorizations) and almost trivially exploit parallelism, first-order methods are an attractive alternative. This article considers fundamental convergence results concerning the application of first-order

methods for the solution of LPs, and proceeds with the many practical considerations that must be made to make first-order methods a viable alternative to standard methods for solving LPs.

In our second article, Shicong Cen and Yuejie Chi provide an optimization-focused exposition to policy gradient methods, a class of algorithms commonly employed in model-free reinforcement learning. The article provides an introduction to fascinating convergence results concerning how (given access to a deterministic expectation of appropriate value functions) various iterative methods based on the projected gradient iteration can identify globally optimal policies in the sense of traditional Markov decision processes. The article then specifies this general framework to show how this class of algorithms can be used to find Nash equilibria in (sequential) matrix games, as well as globally optimal controllers in some common settings of optimal control.

All issues of *Views and News* are available online at <https://siagoptimization.github.io/ViewsAndNews>.

The SIAG on Optimization Views and News mailing list, where editors can be reached for feedback, is siagoptnews@lists.mcs.anl.gov. Suggestions for new issues, comments, and papers are always welcome.

Pietro Belotti

DEIB, Politecnico di Milano

Email: pietro.belotti@polimi.it

Web: <https://belotti.faculty.polimi.it>

Dmitriy Drusvyatskiy

Mathematics Department, University of Washington

Email: ddrusv@uw.edu

Web: <https://sites.math.washington.edu/~ddrusv>

Matt Menickelly

Argonne National Laboratory

Email: mmenickelly@anl.gov

Web: <https://www.mcs.anl.gov/~menickmj>
