Proceedings of Machine Learning Research vol 313:1–52, 2026 37th International Conference on Algorithmic Learning Theory

Reusing Samples in Variance Reduction

Yujia Jin YUJIAJIN@STANFORD.EDU

Ishani Karmarkar ISHANIK@STANFORD.EDU

Aaron Sidford SIDFORD@STANFORD.EDU

Jiayi Wang JYW@STANFORD.EDU

Editors: Matus Telgarsky and Jonathan Ullman

Abstract

We provide a general framework to improve trade-offs between the number of *full batch* and *sample* queries used to solve structured optimization problems. Our results apply to a broad class of randomized optimization algorithms that iteratively solve sub-problems to high accuracy. We show that such algorithms can be modified to *reuse the randomness* used to query the input across sub-problems. Consequently, we improve the trade-off between the number of gradient (full batch) and individual function (sample) queries for finite sum minimization, the number of matrix-vector multiplies (full batch) and random row (sample) queries for top-eigenvector computation, and the number of matrix-vector multiplies with the transition matrix (full batch) and generative model (sample) queries for optimizing Markov Decision Processes. To facilitate our analysis we introduce the notion of *pseudo-independent algorithms*, a generalization of pseudo-deterministic algorithms (Gat and Goldwasser, 2011), that quantifies how independent the output of a randomized algorithm is from a randomness source.

Keywords: Variance reduction, sample reuse, finite-sum minimization, matrix-vector games, Markov decision processes, top-eigenvector computation.

1. Introduction

Variance reduction is a technique for designing efficient algorithms for solving structured optimization problems. This technique consists of reducing the variance of stochastic or randomized access to a component of the input (what we call *sample queries*) by performing occasional, more expensive queries, which involve the entire input (what we call *full batch queries*). Variance reduction (Johnson and Zhang, 2013) has led to improved query complexities for many problems: finite-sum minimization (Frostig et al., 2015; Johnson and Zhang, 2013; Lin et al., 2015), top eigenvector computation (Garber et al., 2016), Markov decision processes (MDPs) (Sidford et al., 2023, 2018a; Jin et al., 2024), and matrix games (Carmon et al., 2019).

Variance reduction schemes often apply an iterative method—which we refer to as an *outer-solver*—to reduce an original problem to solving a sequence of *sub-problems*. The outer-solver runs n_{outer} iterations and, in each iteration, a sub-problem is carefully constructed and solved using what we call a *sub-solver*. This sub-solver is a randomized algorithm that uses n_{batch} full batch and n_{sample} *fresh* sample queries to the input to solve a sub-problem. This solves the orignal problem

with a total of $n_{\text{outer}}n_{\text{batch}}$ full batch and $n_{\text{outer}}n_{\text{sample}}$ sample queries. Often, it is possible to tune the outer loop to trade off how many sub-problems are solved (n_{outer}), with how challenging each sub-problem is to solve (n_{batch} and n_{sample}).

The central question in this work is: *can we improve this trade-off between the number of full batch and sample queries in prominent variance reduction settings?* We ask whether randomness in sample queries can be *reused* across *all* n_{outer} iterations of the outer-solver to reduce the total number of sample queries from $n_{\text{outer}}n_{\text{sample}}$ to just n_{sample}, without sacrificing correctness guarantees. We answer affirmatively by designing a *sample reuse framework* that reuses randomness in variance-reduction settings where (1) the outer-solver is robust to ℓ_∞-bounded random noise in sub-problem solutions, and (2) the sub-solver uses randomness *obliviously*, i.e., sampling a random variable that is independent of the sub-problem. We show that for many structured optimization problems, both properties hold, and consequently the same sample queries *can* be reused across all n_{outer} iterations of the outer-solver.

Applying our sample reuse framework enables us to decrease the total number of sample queries by a factor of n_{outer} for finite-sum minimization, top eigenvector computation, and composite ℓ_2-ℓ_2 matrix games. We also propose a new outer-solver framework for solving *discounted Markov Decision Processes* (DMDPs) that reduces solving a DMDP to solving a sequence of DMDPs of *lower* discount factor. This result—which may be of independent interest—allows us to obtain improved sample query complexities for discounted MDPs and average-reward MDPs as well as faster running times in certain cases. Finally, we show how to apply our framework to obtain sample query complexity improvements for ℓ_2-ℓ_1 matrix games, where prior variance-reduction schemes use a mix of oblivious and non-oblivious sampling.

Organization. This introduction discusses a motivating, illustrative example of finite-sum minimization (Section 1.1), our main results for other structured optimization problems (Section 1.2), and preliminaries (Section 1.3). Section 2, describes our sample-reuse framework and a new notion of *pseudoindependent* algorithms, which generalizes the concept of *pseudodeterminism* introduced in prior work Gat and Goldwasser (2011) and enables our analysis of this sample-reuse framework. Details on how our framework can be applied to many other structured optimization problems can be found in Sections 3 through 7. Section 8 concludes with a summary and some directions for future work. Omitted proofs are in the Appendix.

1.1. Motivating example: convex finite-sum minimization (FSM)

As an illustrative, motivating example, consider the prototypical *finite-sum minimization (FSM problem)* (introduced in detail in Section 3). In the FSM problem, we are given convex, L-smooth functions $f_1, \ldots, f_n : \mathbb{R}^d \to \mathbb{R}$. In the standard query model, we have an oracle that, when queried at $\boldsymbol{x} \in \mathbb{R}^d$ and $i \in [n]$, outputs $\nabla f_i(\boldsymbol{x})$. The goal of FSM is to minimize $F : \mathbb{R}^d \to \mathbb{R}$ defined as $F(\boldsymbol{x}) := \frac{1}{n}\sum_{i\in[n]} f_i(\boldsymbol{x})$ provided that F is μ-strongly convex. (The individual f_i need not be strongly convex.)

Near-optimal query complexities can be obtained for FSM by applying an outer-solver such as accelerated proximal point/Catalyst (APP) (Frostig et al., 2015; Lin et al., 2015) with stochastic variance-reduced gradient descent (SVRG) as the sub-solver (Johnson and Zhang, 2013). APP solves the FSM problem by solving $n_{\text{outer}} = \tilde{O}(\sqrt{\alpha/\mu})$ regularized problems of the form $\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) + \frac{\alpha}{2}\|\boldsymbol{x} - \boldsymbol{x}_t\|_2^2$ where for each iteration $t \in [n_{\text{outer}}]$, $\boldsymbol{x}_t$ depends on the solution to the $(t-1)$-th sub-problem

(Frostig et al., 2015).[1] Each sub-problem is solved using $\tilde{O}(n + L/\alpha)$ queries, e.g., using SVRG as the sub-solver (Johnson and Zhang, 2013). Taking $\alpha = \max\{L/n, \mu\}$ yields a query complexity of $\tilde{O}(n + \sqrt{nL/\mu})$, which is known to be the optimal, up to logarithmic factors (Woodworth and Srebro, 2016; Agarwal and Bottou, 2015).

The batch-sample model. In light of this prior work, to obtain further improvements, we *must go beyond* this standard query model. Consequently, in this paper we consider a more *fine-grained batch-sample query model*, which better captures computational trade-offs that could be present in some settings.

To motivate this model, a closer inspection of the aforementioned algorithm (APP with SVRG) reveals that solving each sub-problem requires $\tilde{O}(1)$ computations of the gradient of F and $\tilde{O}(L/\alpha)$ computations of the gradient of a component f_i, where i is chosen uniformly at random. Using the fact that the a gradient of F can be computed by querying each of the n ∇f_i once (n queries in the standard query model), yields the aforementioned near-optimal query complexity of $\tilde{O}(n + \sqrt{nL/\mu})$ in the standard query model.

However, treating all queries to the gradient of f_i as *computationally equivalent* can obscure the structure of the problem. For example, in some practical computational models, computing the gradient of F could be *much cheaper* than computing ∇f_i for n arbitrary f_i—for example, due to caching behavior or memory layout (Fischetti et al., 2018).[2] Likewise, querying the gradient of the same f_i multiple times could be cheaper than making the same number of queries to different f_i. In such settings, it could be helpful to optimally trade-off between the number of queries to a gradient of F (batch-queries) and the number of queries to a gradient of a random f_i (sample-queries), depending on their relative costs.

Moreover, in some classic problems of FSM—namely, empirical risk minimization— just $O(1)$ queries to the gradient of f_i suffices to *exactly recover* f_i and know ∇f_i at all points *without any* further queries to f_i. This occurs, for example in linear regression when $f_i(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{a}_i^\top \boldsymbol{x} - \boldsymbol{b}(i))^2$ for feature vectors $\boldsymbol{a}_i \in \mathbb{R}^d$ and (explicitly known) labels $\boldsymbol{b} \in \mathbb{R}^n$ and for generalized linear models when $f_i(\boldsymbol{x}) = \phi_i(\boldsymbol{a}_i^\top \boldsymbol{x} - \boldsymbol{b}(i))$ for known ϕ_i (see Section 3.1). Hence, *repeatedly querying the gradient of the same* f_i, say T times, can be much cheaper than querying T arbitrary f_i. For instance, this can be the case in distributed memory layouts Woodworth et al. (2020); Sallinen et al. (2016).

To capture these nuances, we consider a more fine-grained analysis of the complexity of FSM where we allow *smoothly trading off* between two types of queries, depending on their relative costs:

- **Full batch query**: for input $\boldsymbol{x}$ computes $\nabla F(\boldsymbol{x})$, and
- **Sample query**: for input $i \in [n]$ allows $\nabla f_i(\boldsymbol{x})$ to be computed for *any* future input $\boldsymbol{x}$.

For example, recall from above that in the special case of linear regression, each $f_i(x) = \frac{1}{2}(\boldsymbol{a}_i^\top x - \boldsymbol{b}(i))^2$ where $\boldsymbol{a}_i \in \mathbb{R}^d$ is row i of a data matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{b}$ is a label. In this case, a batch oracle

1. As in prior work, throughout, we may use $\tilde{O}(\cdot)$ to hide poly-logarithmic factors in problem parameters.
2. Fischetti et al. (2018) demonstrated empirically that the technique of *minbatch persistency*, which reuses the same minibatch for consecutive SGD iterations, better exploits modern GPUs by reducing the overhead related to data loading and moreover, may lead to faster convergence in practice. Previously, Sallinen et al. (2016) also studied empirical variants of SGD designed to reduce communication in distributed memory stsems and improve cache utilization while preserving statistical efficiency. These empirical findings motivate our theoretical analysis of a more fine-grained batch-sample query model which might better capture such practical considerations.

query is implementable just with the appropriate *matrix-vector multiply*, $\boldsymbol{A}^\top \boldsymbol{A}x$, and a sample query is implementable by outputting the appropriate row, $\boldsymbol{a}_i$ (see Section 3).

This batch-query model captures (1) the fact that full batch queries can be cheaper than n-sample queries due to caching layout as well as (2) the fact that re-querying previously cached components is often cheaper than querying fresh components of F due to caching and communication costs between machines. Depending on the computational environment, one can now seek to optimally *trade-off* these two types of oracle queries, depending on their relative costs. From this perspective, state-of-the-art FSM algorithms consist of $n_{\text{outer}} = \tilde{O}(\sqrt{\alpha/\mu})$ outer loop iterations of the APP outer-solver. The sub-solver in each iteration (SVRG) is implementable with $n_{\text{batch}} = \tilde{O}(1)$ full batch queries and $n_{\text{sample}} = \tilde{O}(L/\alpha)$-sample queries. Tuning α allows us to smoothly trade off between these two types of oracle queries.

As mentioned, prior work established that $\tilde{O}(n + \sqrt{nL/\mu})$-sample queries is near-optimal for FSM if one *only uses sample queries* (Woodworth and Srebro, 2016). This lower bound, when $n = 1$, also implies $\tilde{O}(\sqrt{L/\mu})$ full batch queries is optimal when using *only* full batch queries. Thus, it is perhaps natural to expect that solving FSM with $\tilde{O}(\sqrt{\alpha/\mu})$ full batch queries and $\tilde{O}(L/\sqrt{\mu\alpha})$-sample queries is the optimal query complexity trade-off—after all, it recovers the optimal rate for using *only* full batch queries when $\alpha = L, n = 1$ and the optimal sample complexity for using *only* sample queries if $\alpha = \max(L/n, \mu)$.) Still, we ask: *can this trade-off be improved?*

Our FSM results. We show that this trade-off can be improved! By developing and applying techniques for reusing randomness (Section 2), we provide a method that uses only $\tilde{O}(\sqrt{\alpha/\mu})$-full batch queries and $\tilde{O}(L/\alpha)$-sample queries for any $\alpha \geq \mu$ to solve FSM. That is, we decrease the number of sample queries by $\tilde{O}(n_{\text{outer}}) = \tilde{O}(\sqrt{\alpha/\mu})$. This yields corresponding improvements for regression and generalized linear models (see Section 3.1) and shows these problems can be solved *with less information than was known previously.* Although this doesn't directly yield a worst-case asymptotic-runtime improvement that we are aware of, it sheds new light on the information needed to solve FSM and could yield faster algorithms depending on caching and memory layout.

To obtain this improved trade-off, we observe that in the previously discussed variance-reduction approach, the sub-solver solves each sub-problem to high accuracy by querying the sample oracle for a uniformly random component i. Importantly, this distribution for choosing the i *does not depend* on the particular sub-problem, i.e., the samples are *oblivious* of the point at which the gradients are queried. We refer to such queries as *oblivious sample queries*. As mentioned earlier, we show that by solving the regularized sub-problems to high accuracy and adding a small amount of uniform noise to the output, it is possible to *reuse the same* i in each sub-problem. To facilitate this proof, we introduce a notion of *pseudo-independence* (Section 2), a generalization of pseudo-determinism (Gat and Goldwasser, 2011), and provide general theorems about pseudo-independence in Appendix B.

1.2. Our results for structured optimization

Here we explain our results for prominent variance reduction settings (including FSM). These results follow a similar approach as in Section 1.1, but differ in oracles and sub-problem structure. For each problem we consider using an outer-solver framework (e.g., APP) to iteratively reduce the original optimization problem to a sequence of sub-problems, which are solved to high-accuracy using a sub-solver (e.g., SVRG). (This is depicted by *Outer-Solver* and *Standard Sub-Solver* in Figure 1.)

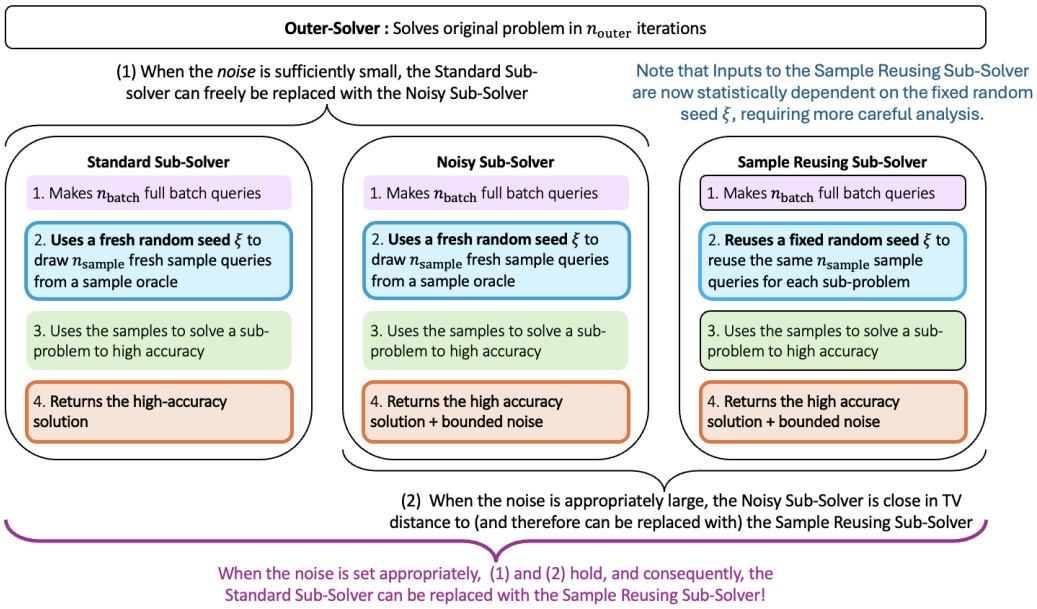

Figure 1: *Sample reuse framework.* The diagram summarizes our approach for replacing Standard Sub-Solvers with a Sample Reusing Sub-Solver, which reuses the same sample queries across all n_{outer} iterations of the Outer-Solver. In (2), TV abbreviates total variation distance. For a more detailed explanation of how the random seed is reused from the inner loop to the outer loop, please see Algorithm 1 and Table 3.

Two observations drive these results. First, in these problems, an outer-solver framework is guaranteed to (probably, approximately) solve the original problem, even if at each iteration, the solution to the t-th sub-problem is perturbed by some bounded random noise. (This is depicted by the *Noisy Sub-Solver* in Figure 1.) Second, when this random noise has a sufficiently large range, the random perturbations protect the randomness of the sample queries well enough so that even if we reuse the *same* sample queries across *all* iterations of the outer-solver (this is depicted by *Sample Reusing Sub-Solver* in Figure 1), the distribution over outputs does not change by much—as measured by total variation (TV) distance. To prove this fact, we describe and analyze a new notion of *pseudo-independence* of randomized algorithms in Section 2.

Combining these observations, we show that in many problems, when the noise is carefully scaled, the Outer-Solver can use the Sample-Reusing Sub-solver as the sub-problem solver (instead of the Standard Sub-Solver). This reduces the total number of sample queries by a multiplicative n_{outer} factor. This is perhaps surprising—it is well-known that randomized algorithms are *not* always amenable to reusing randomness across sequential invocations, see e.g., (Cherapanamjeri and Nelson, 2020; Cohen et al., 2024). However, our analysis shows randomness *can* be reused in many applications of variance reduction.

This sample reuse framework is formalized in Section 2. Although this formalism is somewhat abstract and technical—as it is intended to capture a wide range of variance-reduction settings—it enables us to obtain improved query complexity trade-offs for solving a broad range of prominent optimization and machine-learning problems. These improvements are summarized in Table 1. Note that in certain specialized problems, e.g., TopEV and special cases of FSM such as linear regression, there might be other approaches to obtain our improved trade-offs using preconditioning

(Cohen et al., 2020b, 2015). However, a strength of our sample reuse framework is its versatility—our approach applies even in problems where there is no clear preconditioning analog, e.g., MDPs or matrix games. In the next paragraphs, we describe the implications of our results for several problems beyond FSM as summarized in Table 2.

Markov Decision Processes. In Section 4 we consider the problem of computing an ϵ-optimal policy for a Markov Decision Process (MDP). We denote an MDP by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \boldsymbol{P})$ where $\mathcal{S}$ denotes a finite state space, $\mathcal{A}$ denotes the total set of all state-action pairs, and $\boldsymbol{P}$ denotes the state-action-state transition matrix. We use $\mathcal{A}_{\mathrm{tot}}$ to denote the total number of state-action pairs (see Section 4 for more formal treatment.) We consider two well-studied variants of the problem: γ-discounted MDPs (DMDPs) and average-reward MDPs (AMDPs). In both settings, a *sample query* corresponds to drawing a *random state* from $\boldsymbol{p}(s, a)$ where $\boldsymbol{p}(s, a)$ is the transition probability distribution corresponding to (s, a)-th row of $\boldsymbol{P}$. Meanwhile, a *full batch* query corresponds to a matrix-vector product with the transition matrix $\boldsymbol{P}$.

Prior to our work, the state-of-the-art full batch versus sample query trade-off for DMDPs was obtained by Jin et al. (2024), who provided an algorithm to compute an ϵ-optimal policy for a γ-dscounted DMDP in $\tilde{O}(1)$-full batch queries an $\tilde{O}(\mathcal{A}_{\mathrm{tot}}(1-\gamma)^{-2})$-sample queries for any $\gamma \in (0, 1)$. On the other hand, classic value iteration obtains an ϵ-optimal policy for a γ-discounted DMDP in $\tilde{O}((1-\gamma)^{-1})$-full batch queries and no sample queries. However, note that (in contrast to the *other* structured optimization problems in Table 2) prior work did *not* provide a smooth trade-off between the number of full batch versus sample queries.

To enable a smooth trade-off between the number of full batch and sample queries, we develop a new outer-solver framework which reduces solving a γ-discounted DMDP to solving a sequence of γ'-discounted DMDPs for $\gamma' < \gamma$ (Theorem 35); we believe this result may be of independent interest. Combining our new outer-solver framework for DMDPs with our sample reuse framework, we obtain a *tunable* trade-off between the number of full batch and sample queries needed to solve a γ-DMDP. We show that for any $\alpha \in [1-\gamma, 1)$, there is an algorithm that solves the problem using $\tilde{O}(\alpha(1-\gamma)^{-1})$-full batch queries and $\tilde{O}(\mathcal{A}_{\mathrm{tot}}\alpha^{-2})$-sample queries. Interesting, our new outer-solver framework for DMDPs also yields a new algorithm for solving γ-DMDPs to high-accuracy and improves the *runtime* of prior work (Jin et al., 2024) under appropriate conditions on the *sparsity* of the underlying transition matrix (Theorem 38.) Finally, to extend our results for DMDPs to AMDPs, we leverage a reduction of (Jin and Sidford, 2021).

Problem	Description
Finite-sum minimization (FSM)	$F(x) = \frac{1}{n}\sum_{i\in[n]} f_i(x)$ is a μ-strongly convex function where each f_i is L-smooth and convex. The goal is to reduce the error (with respect to the minimizer of F) of an initial point by a factor of $c > 1$ wp. $1-\delta$. (Section 3 and, in greater generality, Section 6.)
Discounted MDP (DMDP)	The goal is to compute an ϵ-optimal policy for a γ-discounted infinite-horizon MDP wp. $1-\delta$. We use $\mathcal{A}_{\text{tot}}$ to denote the total number of state-action pairs in the MDP and assume that rewards and $\mathcal{A}_{\text{tot}}$ are bounded. (Section 4)
Average-reward MDP (AMDP)	The goal is to compute an ϵ-optimal policy for an average-reward infinite-horizon MDP wp. $1-\delta$. We use $\mathcal{A}_{\text{tot}}$ to denote the total number of state-action pairs in the MDP and assume that rewards and $\mathcal{A}_{\text{tot}}$ are bounded. (Section 4)
Composite ℓ_2-ℓ_2 matrix games (ℓ_2-ℓ_2)	The goal is to compute an ϵ-saddle point for an ℓ_2-ℓ_2 matrix game in matrix $\boldsymbol{A} \in \mathbb{R}^{d\times n}$ wp. $1-\delta$. $\|\boldsymbol{A}\|_F := (\sum_i \sum_j \boldsymbol{A}_{i,j}^2)^{1/2}$ is the Frobenius norm. (Section 5)
Top Eigenvector (TopEV)	The goal is to compute an ϵ-approximate top eigenvector of $\boldsymbol{A}^\top\boldsymbol{A} \in \mathbb{R}^{n\times d}$ with high probability in d. Here $\text{gap}(\boldsymbol{A})$ and $\text{sr}(\boldsymbol{A})$ are the relative eigen-gap and stable rank of $\boldsymbol{A}$ respectively. (Section 7)

Table 1: Summary of structured optimization problems (and abbreviations) studied in this work. Throughout this paper, wp. abbreviates "with probability."

Problem	Prior Work ($\tilde{O}$)			Our Trade-off ($\tilde{O}$)		Range
	FB	OS	Paper	FB	OS	
FSM	$\sqrt{\alpha/\mu}$	$L/\sqrt{\alpha\mu}$	Frostig et al. (2015)	$\sqrt{\alpha/\mu}$	L/α	$\alpha > \mu$
DMDP	1	$\mathcal{A}_{\text{tot}}(1-\gamma)^{-2}$	Jin et al. (2024)	$\alpha(1-\gamma)^{-1}$	$\mathcal{A}_{\text{tot}}\alpha^{-2}$	$1 > \alpha \geq 1-\gamma$
AMDP	1	$\mathcal{A}_{\text{tot}}t_{\text{mix}}^2\epsilon^{-2}$	Jin and Sidford (2021)	$\alpha t_{\text{mix}}\epsilon^{-1}$	$\mathcal{A}_{\text{tot}}\alpha^{-2}$	$1 > \alpha \geq \frac{\epsilon}{9t_{\text{mix}}}$
ℓ_2-ℓ_2	$\alpha\epsilon^{-1}$	$\|\boldsymbol{A}\|_F^2(\alpha\epsilon)^{-1}$	Carmon et al. (2019)	$\alpha\epsilon^{-1}$	$\|\boldsymbol{A}\|_F^2\alpha^{-2}$	$\alpha > 0$
TopEV	$\sqrt{\frac{\alpha}{\text{gap}(\boldsymbol{A})}}$	$\text{sr}(\boldsymbol{A})(\alpha^3\text{gap}(\boldsymbol{A}))^{-\frac{1}{2}}$	Garber et al. (2016)	$\sqrt{\frac{\alpha}{\text{gap}(\boldsymbol{A})}}$	$\text{sr}(\boldsymbol{A})\,\alpha^{-2}$	$\alpha > \Theta(\text{gap}(\boldsymbol{A}))$

Table 2: *Main results*. FB and OS denote the required full batch and oblivious (non-adaptive) sample queries, respectively, with α tuning their trade-off. We compare our trade-offs with prior work. Importantly, the "Our trade-off" column always improves over the trade-off under "Prior Work" under the *same* assumptions and problem definitions made in the prior work.

Matrix games. In Section 5.2 we study the problem of computing an ϵ-Nash equilibrium for a composite ℓ_2-ℓ_2 matrix game, which is the problem of finding an ϵ-approximate saddle point of the following minimax problem: $\min_{\boldsymbol{x}\in\mathbb{B}^n}\max_{\boldsymbol{y}\in\mathbb{B}^m} \boldsymbol{y}^\top\boldsymbol{A}\boldsymbol{x} + \phi(\boldsymbol{x}) - \psi(\boldsymbol{y})$ where $\mathbb{B}^k$ is the k-dimensional Euclidean ball, and ϕ, ψ are (explicitly known) convex functions. In this setting, a *sample* query corresponds to sampling $\boldsymbol{A}_{ij}$ while a *full batch* query corresponds to querying the pair of matrix-vector products $(\boldsymbol{y}^\top\boldsymbol{A}, \boldsymbol{A}\boldsymbol{x})$. Previously, Carmon et al. (2019) showed how to solve

the problem using $\tilde{O}(\alpha\epsilon^{-1})$-full batch queries and $\tilde{O}(\|\boldsymbol{A}\|_F^2(\alpha\epsilon)^{-1})$-sample queries, for any $\alpha > 0$. Using our sample reuse framework, we improve this trade-off and show how to solve the problem using $\tilde{O}(\alpha\epsilon^{-1})$-full batch queries and $\tilde{O}(\|\boldsymbol{A}\|_F^2\alpha^{-2})$-sample queries, for any $\alpha > 0$.

To highlight one interesting application of this, note that for composite ℓ_2-ℓ_2 matrix games, *both* full batch queries and sample queries can be implemented by a (two-sided) matrix-vector oracle which outputs $(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{A}^\top\boldsymbol{y})$ for a $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d \times \mathbb{R}^n$. With $\alpha = \|\boldsymbol{A}\|_F^{2/3}\epsilon^{1/3}$ we obtain an algorithm that solves ℓ_2-ℓ_2 matrix games in only $\tilde{O}(\|\boldsymbol{A}\|_F^{2/3}\epsilon^{-2/3})$-matrix-vector oracle queries. This improves the matrix-vector complexity of the problem over the $\tilde{O}(\|\boldsymbol{A}\|_F\epsilon^{-1})$-matrix-vector oracle queries required in prior work (Carmon et al., 2019; Nemirovski, 2004) and is near-optimal, due to (Liu et al., 2023). *Special* cases of composite ℓ_2-ℓ_2 matrix games reduce to ℓ_2-regression, in which case this trade-off of $\tilde{O}(\|\boldsymbol{A}\|_F^{2/3}\epsilon^{-2/3})$-matrix-vector oracle queries may also follow from preconditioning (Cohen et al., 2020b; Liu et al., 2023). *However*, our work is first to achieve near-optimal matrix-vector oracle query complexity for *general* composite ℓ_2-ℓ_2 matrix games. Subsequent to our work, Karmarkar et al. (2025) recently showed how to match this matrix-vector oracle query complexity deterministically.

In Section 5.2 we also show how to apply our methods to another class of minimax problems, called composite ℓ_2-ℓ_1 matrix games. A composite ℓ_2-ℓ_1 matrix game corresponds to finding an ϵ-approximate saddle point of $\min_{\boldsymbol{x}\in\mathbb{B}^n}\max_{\boldsymbol{y}\in\Delta^m}\boldsymbol{y}^\top\boldsymbol{A}\boldsymbol{x} + \phi(\boldsymbol{x}) - \psi(\boldsymbol{y})$ where Δ^k is the k-dimensional probability simplex and ϕ, ψ are explicitly known convex functions. This setting captures, for example, the problem of computing a hard-margin support vector machine. Notably, in this setting, variance-reduced methods (Carmon et al., 2019) *combine* oblivious sample queries along with non-oblivious sample queries to solve the sub-problems induced by the outer-solver (conceptual proximal point (Nemirovski, 2004; Carmon et al., 2019)). Our sample reuse framework *still* applies and allows us to *reuse* the oblivious (non-adaptive) sample queries across all n_{outer} invocations of the sub-solver. We also discuss how this improves query complexity trade-offs for two computational geometry problems: the minimum enclosing and maximum inscribed ball problems.

Top eigenvector computation. In Section 7 we consider the problem of computing an ϵ-approximate top-eigenevctor for a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. We use $\text{gap}(\boldsymbol{A})$ and $\text{sr}(\boldsymbol{A})$ to denote the relative eigen-gap and stable rank of $\boldsymbol{A}$, respectively. In this setting, a *sample query* corresponds to querying an entry $\boldsymbol{A}_{ij}$, while a *full batch* query corresponds to a matrix-vector product with $\boldsymbol{A}$. Prior work of Garber et al. (2016) showed how to build upon the approximate proximal point methods described for FSM Frostig et al. (2015); Lin et al. (2015) and the *shift-and-invert* power method Saad (2011) to compute an ϵ-approximate top-eigenvector using $\tilde{O}(\sqrt{\alpha/\text{gap}(\boldsymbol{A})})$-full batch queries along with $\tilde{O}(\text{sr}(\boldsymbol{A})(\alpha^3\text{gap}(\boldsymbol{A}))^{-1/2})$-sample queries for any $\alpha > \Theta(\text{gap}(\boldsymbol{A}))$. Using our sample-reuse framework, we improve this trade-off to $\tilde{O}(\sqrt{\alpha/\text{gap}(\boldsymbol{A})})$-full batch queries and $\tilde{O}(\text{sr}(\boldsymbol{A})\alpha^{-2})$-sample queries.

1.3. Preliminaries

General notation. Bold lowercase letters are vectors in $\mathbb{R}^d$ where $\boldsymbol{u}(i)$ is the i-th index of $\boldsymbol{u}$. Bold capital letters are matrices. The ℓ_p norm of $\boldsymbol{u}$ is $\|\boldsymbol{u}\|_p$. For random variable A over the probability space $(\Omega, \mathcal{F})$, p_A is its probability measure. For event E, $p_{A|E}$ is the measure of A conditioned on E, $\neg E$ is the complement of E, and $\mathbb{1}[E]$ is the indicator of E. For random variables A, B, $A \overset{\mathcal{D}}{=} B$ denotes that A and B are equal in distribution. For measures p and q, $d_{TV}(p, q) := \max_{E\in\mathcal{F}}|p(E) - p(E)|$ is the total variation (TV) distance between p and q. The

support of distribution $\mathcal{D}$ or measure q is denoted $\mathsf{supp}(\mathcal{D})$ or $\mathsf{supp}(q)$. Ber and Unif respectively denote Bernoulli and uniform distributions. $\mathsf{Unif}^d(a,b)$ denotes a d-dimensional vector in which the i-th entry is an independently and identically distributed $\mathsf{Unif}(a,b)$ random variable.

Randomized algorithm notation. We consider algorithms that use up to two independent sources of randomness. We use $\mathcal{A}_{\xi,\chi}$ to denote a randomized algorithm that takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and independent random seeds $\xi \sim \mathcal{D}_\xi$ and $\chi \sim \mathcal{D}_\chi^{\boldsymbol{u}}$ where $\mathcal{D}_\xi$ is *independent of, or oblivious with respect to,* $\boldsymbol{u}$ and $\mathcal{D}_\chi^{\boldsymbol{u}}$ might be *dependent on, or be adaptive with respect to* $\boldsymbol{u}$. We also use the notation $\mathsf{D}_\chi = \{\mathcal{D}_\chi^{\boldsymbol{u}}\}_{\boldsymbol{u}\in\mathbb{R}^d}$ to denote the family of distributions parameterized by $\boldsymbol{u} \in \mathbb{R}^d$. On input $\boldsymbol{u} \in \mathbb{R}^d$, $\mathcal{A}_{\xi,\chi}$ outputs $\mathcal{A}_{\xi,\chi}(\boldsymbol{u})$ for randomly drawn $\xi \sim \mathcal{D}_\xi$ and $\chi \sim \mathcal{D}_\chi^{\boldsymbol{u}}$. At times, we analyze the output of $\mathcal{A}_{\xi,\chi}$ *conditioned* on the value of one or both random seeds. Specifically, $\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u})$ and $\mathcal{A}_{\xi,\chi=c}(\boldsymbol{u})$ are used to denote the randomized algorithms obtained by fixing the value of $\xi = s \in \mathsf{supp}(\mathcal{D}_\xi)$ or $\chi = c \in \mathsf{supp}(\mathcal{D}_\chi^{\boldsymbol{u}})$, respectively. Analogously, $\mathcal{A}_{\xi=s,\chi=c}$ is a *deterministic* algorithm corresponding to conditioning on $\xi = s \in \mathsf{supp}(\mathcal{D}_\xi)$ and $\chi = c \in \mathsf{supp}(\mathcal{D}_\chi^{\boldsymbol{u}})$. We occasionally specify a decomposition of the seed χ into two sub-seeds $\chi = (\nu, \iota)$ where sub-seeds ν, ι are drawn independently from $\nu \sim \mathcal{D}_\nu^x$ and $\iota \sim \mathcal{D}_\iota$. We assume algorithms that output vectors in $\mathbb{R}^d$ have runtime $\Omega(d)$. We use *high-accuracy* to refer to families of algorithms (parameterized by error and failure probability parameters) with at most *polylogarithmic* dependence on their accuracy and failure probability.

2. Sample reuse framework

Here we provide, contextualize, and analyze our sample-reuse framework. First, in Section 2.1 we introduce the framework, the main theorem regarding it (Theorem 6) and formally define *psuedoindependence* (Definition 5) which we use to prove the theorem. For additional context, we then compare pseudo-independence to related definitions in prior work in Section 2.2. We then conclude this section in Section 2.3 by analyzing the sample-reuse framework and proving Theorem 6.

2.1. Sample-reuse framework

Here, we introduce our sample-reuse framework. To describe the framework we provide a general *Meta-Algorithm* for solving an optimization problem. This Meta-Algorithm encompasses three different templates for variance reduction methods. Our framework relates these different templates and applying this relationship to different algorithms yields our improved query-complexity tradeoffs.

More formally, the Meta-Algorithm 1 iteratively reduces solving a problem instance X to solving a sequence of sub-problems. First, the algorithm initializes $\boldsymbol{u}_0 \in \mathbb{R}^d$; an oblivious distribution $\mathcal{D}_\xi$ for sampling a random seed ξ; and a *family* of *non-oblivious* distributions $\mathsf{D}_\chi = \{\mathcal{D}_\chi^x\}_{x\in\mathbb{R}^d}$. It iteratively runs one of *three* possible sub-routines (corresponding to the three different templates), which we describe below, and outputs a convex combination of the iterates. Table 3 summarizes the notation.

The standard sub-routine. The first template is $\texttt{Outer}(X; \texttt{Standard})$, i.e., Meta-Algorithm 1 with sub-routine `StandardLoop` (in the prose, we omit the τ since it has no effect when $\texttt{Type} = \texttt{Standard}$.) This template formalizes the "Standard Sub-Solver" panel in Figure 1 and describes the standard variance reduction template that applies to many algorithms for structured optimization problems—including those cited under "Prior Work" in Table 2. In $\texttt{Outer}(X; \texttt{Standard})$, the for

loop (Line 1) iteratively reduces the original optimization problem to solving n_{outer} sub-problems, where the t-th sub-problem is determined by the $(t-1)$-th iterate, $\boldsymbol{u}_{t-1}$.

The sub-problem solver, or *sub-solver*, denoted $\mathcal{A}^{\text{sub}}_{\xi,\chi}$, is a randomized algorithm that solves the sub-problem at each iteration by (1) making some deterministic full batch queries to the full batch oracle; (2) using the random seed ξ to make randomized *oblivious* sample queries to a sample oracle; and (3) using the random seed χ to perform some additional randomization. Depending on the application (Table 1), (3) might involve adding some noise to the sub-problem or making additional *adaptive* sample queries to a sample oracle. $\mathcal{A}^{\text{sub}}_{\xi,\chi}$ then outputs an intermediate iterate $\boldsymbol{u}_{t-1/2} \in \mathbb{R}^p$. At the end of each iteration, the routine then applies a "post-process" ζ which performs some *deterministic* post-processing of the pair $(\boldsymbol{u}_t, \boldsymbol{u}_{t-1/2})$ (e.g., implementing acceleration/momentum.)

As a illustrative, concrete, example, consider the motivating example of FSM (Section 1.1). Meta-Algorithm 1 captures the previously described combination of APP and SVRG to solve FSM (Frostig et al., 2015; Lin et al., 2015). That is, APP reduces the original problem to solving a sequence of regularized sub-problems, while $\mathcal{A}^{\text{sub}}_{\xi,\chi}$ represents the sub-solver SVRG (Johnson and Zhang, 2013), which is used to solve the regularized sub-problems. In this case, ζ implements acceleration as in (Frostig et al., 2015), and $\boldsymbol{w}$ is set such that the algorithm will simply return the last iterate, $\boldsymbol{u}_{n_{\text{outer}}}$. (See also Section 3.)

The noisy sub-routine. To motivate the second template, recall our motivating observation that for many classes of structured optimization problems (Table 1), there is an instantiation of $\texttt{Outer}(X; \texttt{Standard})$ that provably solves the problem instance X *provided* that: at each iteration $t \in [n_{\text{outer}}]$, the sub-solver $\mathcal{A}^{\text{sub}}_{\xi,\chi}$ solves the sub-problem induced by $\boldsymbol{u}_{t-1}$ to *high-accuracy* in the ℓ_∞ norm. We formalize this observation with the following two definitions.

Definition 1 (Function approximation) *We say that a randomized algorithm $\mathcal{A}_{\xi,\chi}$ is an (η, δ)-*approximation *of $f : \mathbb{R}^d \to \mathbb{R}^p$ if $\mathcal{A}_{\xi,\chi}$ takes an input $\boldsymbol{u} \in \mathbb{R}^d$ along with two random seeds $\xi \sim \mathcal{D}_\xi$ and $\chi \sim \mathcal{D}^{\boldsymbol{u}}_\chi$ and outputs $\mathcal{A}_{\xi,\chi}(\boldsymbol{u}) \in \mathbb{R}^p$ such that $\mathbb{P}_{\xi\sim\mathcal{D}_\xi, \chi\sim\mathcal{D}^{\boldsymbol{u}}_\chi}\left(\|\mathcal{A}_{\xi,\chi}(\boldsymbol{u}) - f(\boldsymbol{u})\|_\infty \le \eta\right) \ge 1 - \delta$.*

Definition 2 (ℓ_∞-robust) *We say* $\texttt{Outer}(X; \texttt{Standard})$ *is* (η, β)-robust *with respect to f^{sub} if there exists a deterministic function $f^{\text{sub}} : \mathbb{R}^d \to \mathbb{R}^p$ such that* $\texttt{Outer}(X; \texttt{Standard})$ *is guaranteed to output a β-accurate solution for X* whenever, *for all $t \in [n_{\text{outer}}]$, $\|\boldsymbol{u}_{t-1/2} - f^{\text{sub}}(\boldsymbol{u}_{t-1})\|_\infty \le \eta$.*

To interpret these definitions, suppose that $\texttt{Outer}(X; \texttt{Standard})$ is (η, β)-*robust*, $\tau \le \eta/2$, and the sub-solver $\mathcal{A}^{\text{sub}}_{\xi,\chi}$ is an $(\eta/2, \delta)$-approximation of f^{sub}. Then *even if*, at every iteration $t \in [n_{\text{outer}}]$, $\boldsymbol{u}_{t-1/2}$ were to be randomly perturbed by some τ-bounded random noise, the output would *still* be a β-accurate solution to X wp. $1 - n_{\text{outer}}\delta$.

Correspondingly our second sub-routine $\texttt{NoisyLoop}_\tau$ in Meta-Algorithm 1 (and the corresponding template $\texttt{Outer}(X; \texttt{Noisy})$) is *identical* to $\texttt{StandardLoop}$ (correspondingly, the algorithmic template $\texttt{Outer}(X; \texttt{Standard})$) *except* that at each iteration, $\boldsymbol{u}_{t-1/2}$ is *perturbed* by a small amount of $\mathsf{Unif}^p(-\tau, \tau)$ random noise. This formalizes the "Noisy Sub-Solver" in Figure 1. Following the previous intuition, the following lemma shows that when τ is sufficiently small, we can freely interchange $\texttt{Outer}(X; \texttt{Standard})$ with $\texttt{Outer}(X; \texttt{Noisy}, \tau)$, without losing correctness guarantees.

Lemma 3 *Suppose* $\texttt{Outer}(X; \texttt{Standard})$ *is (η, β)-robust with respect to f^{sub}, $\tau \in (0, \eta/2)$, and $\mathcal{A}^{\text{sub}}_{\xi,\chi}$ is an $(\eta/2, \delta)$-approximation of f^{sub}. Then* $\texttt{Outer}(X; \texttt{Standard})$ *and* $\texttt{Outer}(X; \texttt{Noisy}, \tau)$*, wp. $1 - n_{\text{outer}}\delta$, output a β-accurate solution to X.*

Table 3: Parameter table for Meta Algorithm (Algorithm 1).

Symbol	Description
$\mathcal{D}_\xi$	This is an oblivious distribution governing the random variable ξ.
D_χ	This is a family of distributions parameterized by $\boldsymbol{x} \in \mathbb{R}^d$. It governs the *adaptive* random variable χ. We use $\mathcal{D}_\chi^{\boldsymbol{x}}$ to denote the distribution in D_χ corresponding to an $\boldsymbol{x} \in \mathbb{R}^d$.
$\mathcal{A}^{\text{sub}}_{\xi,\chi}$	$\mathcal{A}^{\text{sub}}_{\xi,\chi}$ is a sub-problem solver (sub-solver) that takes in $\boldsymbol{u} \in \mathbb{R}^d$ and seeds ξ and χ and outputs $\mathcal{A}^{\text{sub}}_{\xi,\chi}(\boldsymbol{u}) \in \mathbb{R}^p$. $\mathcal{A}^{\text{sub}}_{\xi,\chi}$ may make full batch queries and sample queries to X. Batch queries depend *only* on $\boldsymbol{u}$. The seed ξ is used to make *oblivious* sample queries. Meanwhile, the seed χ may optionally be used to make additional *adaptive* sample queries.
ζ	$\zeta : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ is a deterministic post-processing function; we call it an *outer-process* since it captures the role of the *outer-solver* discussed in Section 1 (Figure 1).
$\boldsymbol{w}$	The algorithm outputs a convex combination of the $\boldsymbol{u}_t$ given by coefficient vector $\boldsymbol{w}$.
τ	Noise parameter $\tau > 0$ controls the amount of noise to be added.

Proof The proof is immediate from Definition 2 and union bound over all n_{outer} iterations. ■

Algorithm 1: Variance-Reduction Meta-Algorithm $\texttt{Outer}(X; \texttt{Type}, \tau)$

Parameter: Loop specification: $\texttt{Type} \in \{\texttt{Standard}, \texttt{Noisy}, \texttt{Reuse}\}$ // Types defined below

Initialize $\boldsymbol{u}_0 \in \mathbb{R}^d$

// If $S = \texttt{Standard}$, τ may be omitted, in which case Line 1 is skipped and τ is omitted in Line 1

// The next line builds a noisy version of $\mathcal{A}^{\text{sub}}_{\xi,\chi}$, which is denoted $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$

Let $\mathcal{D}_{\chi'}^{\boldsymbol{u}_{t-1}} := \mathcal{D}_{\chi}^{\boldsymbol{u}_{t-1}} \times \mathsf{Unif}^p(-\tau, \tau)$ and $\mathcal{A}'^{\text{sub}}_{\xi=s,\chi'=(c,\boldsymbol{e})}(\boldsymbol{u}_{t-1}) := \mathcal{A}^{\text{sub}}_{\xi=s,\chi=c}(\boldsymbol{u}_{t-1}) + \boldsymbol{e}$

Draw $s_1, ..., s_{n_{\text{outer}}} \sim \mathcal{D}_\xi$ // Draw seeds from the oblivious distribution

for *each* $t \in [n_{\text{outer}}]$ **do call** $\langle \texttt{Type} \rangle \texttt{Loop}_\tau$

return: $\sum_{t \in [n_{\text{outer}}]} \boldsymbol{w}(t) \cdot \boldsymbol{u}_t$

// Different loops that can be called on Line 1

$\texttt{StandardLoop}_\tau$**:** Draw $c_t \sim \mathcal{D}_{\chi}^{\boldsymbol{u}_{t-1}}$, $\boldsymbol{u}_{t-\frac{1}{2}} \leftarrow \mathcal{A}^{\text{sub}}_{\xi=s_t,\chi=c_t}(\boldsymbol{u}_{t-1})$, $\boldsymbol{u}_t \leftarrow \zeta(\boldsymbol{u}_{t-1}, \boldsymbol{u}_{t-\frac{1}{2}})$

$\texttt{NoisyLoop}_\tau$**:** Draw $(c_t, \boldsymbol{e}_t) \sim \mathcal{D}_{\chi'}^{\boldsymbol{u}_{t-1}}$, $\boldsymbol{u}_{t-\frac{1}{2}} \leftarrow \mathcal{A}'^{\text{sub}}_{\xi=s_t,\chi'=(c_t,\boldsymbol{e}_t)}(\boldsymbol{u}_{t-1})$,
$\boldsymbol{u}_t \leftarrow \zeta(\boldsymbol{u}_{t-1}, \boldsymbol{u}_{t-\frac{1}{2}})$

$\texttt{ReuseLoop}_\tau$**:** Draw $(c_t, \boldsymbol{e}_t) \sim \mathcal{D}_{\chi'}^{\boldsymbol{u}_{t-1}}$, $\boldsymbol{u}_{t-\frac{1}{2}} \leftarrow \mathcal{A}'^{\text{sub}}_{\xi=s_1,\chi'=(c_t,\boldsymbol{e}_t)}(\boldsymbol{u}_{t-1})$,
$\boldsymbol{u}_t \leftarrow \zeta(\boldsymbol{u}_{t-1}, \boldsymbol{u}_{t-\frac{1}{2}})$

The sample-reuse sub-routine. At this point, it is perhaps natural to wonder: *why* is it helpful to work with the $\texttt{NoisyLoop}_\tau$ subroutine as opposed to the standard, $\texttt{StandardLoop}$ sub-routine in Meta-Algorithm 1? After all, for any τ, both $\texttt{StandardLoop}$ and $\texttt{NoisyLoop}_\tau$ require the same number of batch and sample queries, so there is no obvious computational advantage to using $\texttt{NoisyLoop}_\tau$.

As discussed in our second observation in Section 1.2, the key idea is the following: the random noise at each iteration ensures that the iterates $\boldsymbol{u}_t$ in $\texttt{NoisyLoop}_\tau$ are *almost independent* of the randomness in the sample queries induced by $\xi \sim \mathcal{D}_\xi$ in the following sense: *even if we were to reuse the same realization of* ξ in all iterations $t \in [n_{\text{outer}}]$ of $\texttt{NoisyLoop}_\tau$, the returned output would be close—in total variation (TV) distance—to the output of $\texttt{Outer}(X; \texttt{NoisyLoop}, \tau)$. This brings us to our third and final sub-routine.

The third and final sub-routine $\texttt{ReuseLoop}_\tau$ in Meta-Algorithm 1 (and its corresponding algorithmic template $\texttt{Outer}(X; \texttt{Reuse})$) is *identical* to the $\texttt{NoisyLoop}_\tau$ sub-routine (correspondingly, the algorithmic template $\texttt{Outer}(X; \texttt{Noisy})$) *except that* $\texttt{ReuseLoop}_\tau$ *reuses* a single realization $s_1 \sim \mathcal{D}_\xi$ in all iterations $t \in [n_{\text{outer}}]$. This formalizes the "Sample Reusing Sub-Solver" in Figure 1. Reusing this realization across all iterations corresponds to reusing oblivious sample queries to X. Consequently, for any $\tau > 0$, $\texttt{Outer}(X; \texttt{Reuse}, \tau)$ has lower sample query complexity (by an n_{outer} multiplicative factor) than the original $\texttt{Outer}(X; \texttt{Noisy}, \tau)$ or $\texttt{Outer}(X; \texttt{Standard})$.

Pseudo-independence. To obtain improved query-complexity tradeoffs, our goal is to derive conditions when $\texttt{Outer}(X; \texttt{Reuse}, \tau)$ can be used in place of $\texttt{Outer}(X; \texttt{Noisy}, \tau)$—without sacrificing correctness. We achieve this goal by introducing a new notion of *pseudo-independence*, which we use to bound the total variation (TV) distance between the outputs of $\texttt{Outer}(X; \texttt{Noisy}, \tau)$ and $\texttt{Outer}(X; \texttt{Reuse}, \tau)$. Indeed, if this TV distance is small then as long as $\texttt{Outer}(X; \texttt{Noisy}, \tau)$ is correct, with high probability, so is $\texttt{Outer}(X; \texttt{Reuse}, \tau)$. To define pseudo-independence we first define a *smoothing* of a randomized algorithm as follows.

Definition 4 ((ϵ, δ)-smoothing) *Let $\mathcal{A}_{\xi,\chi}$ be a randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and two random seeds $\xi \sim \mathcal{D}_\xi, \chi \sim \mathcal{D}_\chi^{\boldsymbol{u}}$. We say that algorithm $\bar{\mathcal{A}}_\chi$ is an* (ϵ, δ)-smoothing of $\mathcal{A}_{\xi,\chi}$ with respect to ξ *if it takes as an input $\boldsymbol{u} \in \mathbb{R}^d$ along with one random seed $\chi \sim \mathcal{D}_\chi^{\boldsymbol{u}}$ and for all $\boldsymbol{u} \in \mathbb{R}^d$, $\mathbb{P}_{s\sim\mathcal{D}_\xi}[d_{TV}(p_{\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u})}, p_{\bar{\mathcal{A}}_\chi(\boldsymbol{u})}) \le \epsilon] \ge 1 - \delta$.*

We say a randomized algorithm is (ϵ, δ)-pseudo-independent if we can guarantee *existence* of an (ϵ, δ)-smoothing. Importantly, this smoothing need not be implementable or even explicitly known.

Definition 5 ((ϵ, δ)-pseudo-independence) *Let $\mathcal{A}_{\xi,\chi}$ be as in Definition 4. $\mathcal{A}_{\xi,\chi}$ is* (ϵ, δ)-pseudo-independent of ξ *if it admits an (ϵ, δ)-smoothing with respect to ξ.*

Intuitively, $\mathcal{A}_{\xi,\chi}$ is (ϵ, δ)-pseudo-independent of ξ if, wp. $1-\delta$ over the draw of $s \sim \mathcal{D}_\xi$, $\mathcal{A}_{\xi,\chi}$ is *almost* independent of the first source of randomness—in the sense that wp. $1-\epsilon$ over the draw of χ, $\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u})$ is equal in distribution to a random variable that is *independent* of ξ (see also Fact 7 for further discussion.)

We use this notion of pseudo-independence to bound the TV distance between $\texttt{Outer}(X; \texttt{Noisy}, \tau)$ and $\texttt{Outer}(X; \texttt{Standard})$. First, in Appendix A we show that when the noise parameter τ is set appropriately the sub-solver algorithm $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$ in Meta-Algorithm 1 is (ϵ, δ)-pseudo-independent of ξ. Intuitively, this means that $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$ is *almost independent* of ξ and consequently, it should be possible to reuse the same realization of the seed ξ in each invocation of $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$.

To formalize this, we observe that $\texttt{NoisyLoop}_\tau$ repeatedly composes ζ with $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$ where $\xi_t \sim \mathcal{D}_\xi$ is drawn fresh in each iteration $t \in [n_{\text{outer}}]$. Meanwhile, $\texttt{ReuseLoop}_\tau$ repeatedly composes ζ with $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$ where the same realization $s_1 \sim \mathcal{X}_\xi$ is *reused* in each iteration $t \in [n_{\text{outer}}]$. In

Appendix B we provide a general theorems about pseueodindependence, which allow us to leverage the fact that $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$ is (ϵ, δ)-pseudoindepent of ξ to bound the TV distance between these repeated compositions as a function of $n_{\text{outer}}, \delta, \epsilon$. Thus, when τ, ϵ, δ are set appropriately, the TV distance between the outputs of $\texttt{Outer}(X; \texttt{Noisy})$ and $\texttt{Outer}(X; \texttt{Reuse})$ is small. This TV distance bound allows us to prove the following theorem, which in turn yields all the results in Table 2.

Theorem 6 *Suppose* $\texttt{Outer}(X; \texttt{Standard})$ *is* (η, β)*-robust with respect to* f^{sub} *and* $\delta \in (0, 1)$*. Let* $\eta' := \min(\eta/2, \eta\delta)$*. Suppose* $\mathcal{A}^{\text{sub}}_{\xi,\chi}$ *is an* (η', δ)*-approximation of* f^{sub}*. Then we have that wp.* $1 - 5n^2_{\text{outer}}\delta$*,* $\texttt{Outer}(X; \texttt{Reuse}, \eta'/(2\delta))$ *outputs a* β*-accurate solution to* X*.*

In the problems described in Table 2, the standard variance-reduced algorithms are instantiations of $\texttt{Outer}(X; \texttt{Standard})$ and (η, β)-robust, for η scaling polynomially in β and the problem parameters. This ensures that the blowups in error and failure probability in Theorem 6 are at most *polylogarithmic* in all problem parameters. Moreover, for all of our applications, the sub-solvers (e.g., SVRG) are *high-accuracy* algorithms, meaning that they have query- and time-complexity *polylogarithmic* in the accuracy and failure probability parameters. Thus, the polynomial blow-ups in accuracy and failure probability in Theorem 6 imply at most *polylogarithmic* growth in complexity.

2.2. Comparisons of pseudoindependence to prior work

In this section, we discuss *pseudo-independence*, which generalizes the well-studied notion of pseudo-determinism (Definition 8) (Goldwasser et al., 2017; Goldwasser and Grossman, 2017; Braverman et al., 2023; Grossman et al., 2023; Dixon et al., 2022) and is also related to *reproducibility* (Impagliazzo et al., 2022). First, we state the following Fact 7, which will be helpful in our analysis.

Fact 7 (Lemma 4.1.13 of (Roch, 2024), restated) *Let* $\epsilon > 0$ *and* A *and* B *be random variables. Then* $d_{TV}(p_A, p_B) \leq \epsilon$ *if and only if there exist random variables* C, D, F *and an independent event* E *such that* $\mathbb{P}\{E\} = 1 - \epsilon$*;* $A \overset{\mathcal{D}}{=} C\mathbb{1}\mathcal{E} + D\mathbb{1}\neg\mathcal{E}$*; and* $B \overset{\mathcal{D}}{=} C\mathbb{1}\mathcal{E} + F\mathbb{1}\neg\mathcal{E}$*.*

This fact implies the following intuitive interpretation of pseudo-independence. $\mathcal{A}_{\xi,\chi}$ is (ϵ, δ)-pseudo-independent of ξ if wp. $1 - \delta$ over the draw of $s \sim \mathcal{D}_\xi$, $\mathcal{A}_{\xi,\chi}$ is *almost* independent of the first source of randomness—in the sense that wp. $1 - \epsilon$ over the draw of χ, $\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u})$ is equal in distribution to a random variable that is *independent* of ξ.

Now, we briefly compare pseudo-independence to pseudo-determinism and reproducibility. The term *pseudo-deterministic algorithm*—introduced by Gat and Goldwasser (2011)—describes an algorithm that outputs a deterministic value with high probability. This is formalized with the following definition.

Definition 8 (δ-pseudo-deterministic algorithm) *Let* $\mathcal{A}_\xi$ *be a randomized algorithm which takes an input* $\boldsymbol{u} \in \mathbb{R}^d$ *and a random seed* $\xi \sim \mathcal{D}_\xi$*.* $\mathcal{A}_\xi$ *is* δ-pseudo-deterministic *if there exists a function* h *over* $\mathbb{R}^d$ *such that* $\mathbb{P}_{s\sim\mathcal{D}_\xi}\{\mathcal{A}_{\xi=s}(\boldsymbol{u}) = h(\boldsymbol{u})\} \geq 1 - \delta$*.*

Intuitively, in a δ-pseudo-deterministic algorithm $\mathcal{A}_\xi$, the role of h is similar to that of a smoothing (Definition 4) in the definition of pseudo-independence (Definition 5); however, h must be deterministic whereas as a smoothing can be a randomized algorithm. To formalize this intuition we

need to introduce some additional notation because pseudo-determinism is defined in terms of a single source of randomness, whereas pseudo-independence (Definition 5) is defined in terms of two sources of randomness. Thus, to compare pseudo-determinism to pseudo-independence, we define a simple way to *lift* a single-seed randomized algorithm $\mathcal{A}_\xi$ to two sources of randomness.

Definition 9 *Let $\mathcal{A}_\xi$ be a randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and a random seed $\xi \sim \mathcal{D}_\xi$. Let $\mathcal{A}^{\mathrm{pd}}_{\xi,\chi}(\boldsymbol{u})$ be the randomized algorithm which takes input $\boldsymbol{u} \in \mathbb{R}^d$ and seeds $\xi \sim \mathcal{D}_\xi$ and $\chi \sim \mathcal{D}^x_\chi$ where $\mathsf{supp}(\mathcal{D}^x_\chi) = \{0\}$; and maps $\boldsymbol{u} \mapsto \mathcal{A}_\xi(\boldsymbol{u})$.*

That is, $\mathcal{A}^{\mathrm{pd}}_{\xi,\chi}$ is identical to $\mathcal{A}_\xi$; however, it accepts an additional 0-bit "dummy" random seed χ. The next lemma explains how pseudo-independence captures pseudo-determinism as a special case.

Lemma 10 *$\mathcal{A}_\xi$ is δ-pseudo-deterministic if and only if $\mathcal{A}^{\mathrm{pd}}_{\xi,\chi}$ is $(0,\delta)$-pseudoindependent of ξ.*

Proof First, suppose that $\mathcal{A}_\xi$ is δ-pseudo-deterministic. Then, there exists a function h such that

$$1 - \delta \geq \mathbb{P}_{s\sim\mathcal{D}_\xi}\left\{\mathcal{A}_{\xi=s}(\boldsymbol{u}) = h(\boldsymbol{u})\right\} = \mathbb{P}_{s\sim\mathcal{D}_\xi}\left\{d_{TV}\left(p_{\mathcal{A}^{\mathrm{pd}}_{\xi=s,\chi}(\boldsymbol{u})}, p_{h(\boldsymbol{u})}\right) = 0\right\}.$$

Hence, if $\mathcal{A}'_\chi$ is the algorithm that deterministically maps $\boldsymbol{u} \mapsto h(\boldsymbol{u})$ then $\mathcal{A}'_\chi$ is a $(0,\delta)$-smoothing of $\mathcal{A}^{\mathrm{pd}}_{\xi,\chi}$ with respect to ξ. Thus, $\mathcal{A}^{\mathrm{pd}}_{\xi,\chi}$ is $(0,\delta)$- pseudo-independent with respect to ξ.

Conversely, suppose that $\mathcal{A}^{\mathrm{pd}}_{\xi,\chi}$ is $(0,\delta)$-pseudo-independent with respect to ξ and let $\mathcal{A}'_\chi$ be a $(0,\delta)$-smoothing of $\mathcal{A}^{\mathrm{pd}}_{\xi,\chi}$ with respect to ξ. Then,

$$1 - \delta \geq \mathbb{P}_{s\sim\mathcal{D}_\xi}\left\{d_{TV}\left(p_{\mathcal{A}^{\mathrm{pd}}_{\xi=s,\chi}(\boldsymbol{u})}, p_{\mathcal{A}'_\chi(\boldsymbol{u})}\right) = 0\right\} = \mathbb{P}_{s\sim\mathcal{D}_\xi}\left\{\mathcal{A}_{\xi=s}(\boldsymbol{u}) = \mathcal{A}'_{\chi=0}(\boldsymbol{u})\right\}.$$

Letting h to be the function mapping $\boldsymbol{u} \mapsto \mathcal{A}'_{\chi=0}(\boldsymbol{u})$, we see that $\mathcal{A}_\xi$ is δ-pseudo-deterministic. ■

On the other hand, *reproducibility* is a related notion introduced in Impagliazzo et al. (2022).

Definition 11 ($(\delta,\mathcal{D})$-reproducibile algorithm) *Let $\mathcal{A}_\xi$ be a randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and a random seed $\xi \sim \mathcal{D}_\xi$. Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$. $\mathcal{A}_\xi$ is a $(\delta,\mathcal{D})$-*reproducible algorithm *if there is a function h over $\mathsf{supp}(\mathcal{D}_\xi)$ such that $\mathbb{P}_{\boldsymbol{u}\sim\mathcal{D},s\sim\mathcal{D}_\xi}\left\{\mathcal{A}_{\xi=s}(\boldsymbol{u}) = h(s)\right\} \geq 1 - \delta$.*

Reproducibility asks that with high probability over the draw of an input sample $\boldsymbol{u} \sim \mathcal{D}$ *and* of the random seed $s \sim \mathcal{D}_\xi$, the algorithm outputs a deterministic function of the realized seed: $h(s)$. On the other hand, pseudo-determinism asks that for *every* input $\boldsymbol{u}$, with high probability over the random draw $s \sim \mathcal{D}_\xi$, the randomized algorithm outputs a deterministic function of the input $h(\boldsymbol{u})$. Finally, pseudoindependence asks that for *every* input $\boldsymbol{u}$, a randomized algorithm should be *almost independent* of one of its random seeds in the sense that with high probability over the draw of $s \sim \xi$, its output is close in total-variation distance to a randomized algorithm which is completely oblivious of s.

2.3. Analysis of the sample-reuse framework

In this section, we given an outline of our proof of Theorem 6. Our discussion here expands on the intuitive explanation in Section 2.1. To reduce notational clutter, throughout this section, we omit the superscript sub from $\mathcal{A}'^{\text{sub}}_{\xi,\chi'}$ and refer to it just as $\mathcal{A}'_{\xi,\chi'}$.

Inducing pseudoindependence in the noisy solver. Recall that in Section 2.1, we said that the first step in our proof of Theorem 6 would be to show that $\mathcal{A}'_{\xi,\chi'}$ in Meta-Algorithm 1 is pseudo-independent of ξ when the noise parameter τ is set appropriately. Concretely, we prove the following theorem in Appendix A.

Theorem 12 *Let $\epsilon, \delta \in (0,1)$, $\eta > 0$, $\eta' = \min(\eta/2, \eta\epsilon/p)$ and let $\mathcal{A}_{\xi,\chi}$ be a randomized algorithm that is an $(\eta/2, \delta)$-approximation of a function $f : \mathbb{R}^d \to \mathbb{R}^p$. Let $\mathcal{A}'_{\xi,\chi'}$ be the randomized algorithm defined as follows. $\mathcal{A}'_{\xi,\chi'}$ takes input $\boldsymbol{u} \in \mathbb{R}^d$, random seed $\xi \sim \mathcal{D}_\xi$, and χ' where $\chi' = (\chi, \boldsymbol{\nu}) \sim \mathcal{D}^{\boldsymbol{u}}_{\chi'}$ is the concatenation of an independently drawn seed $\chi \sim \mathcal{D}^{\boldsymbol{u}}_{\chi}$ and seed $\boldsymbol{\nu} \sim \mathcal{D}_\nu := \mathsf{Unif}^p(-\tau, \tau)$ for $\tau := \max\left(\frac{\eta'}{2}, \frac{\eta' p}{2\epsilon}\right)$. For any realization $s, c, \boldsymbol{e}$ of $\xi, \chi, \boldsymbol{\nu}$ and any $\boldsymbol{u} \in \mathbb{R}^d$, we define $\mathcal{A}'_{\xi=s,\chi'=(c,\boldsymbol{e})}(\boldsymbol{u}) = \mathcal{A}_{\xi=s,\chi=c}(\boldsymbol{u}) + \boldsymbol{e}$. Then, $\mathcal{A}'_{\xi,\chi'}$ is an (ϵ, δ)-pseudo-independent of ξ and an (η, δ)-approximation of f with the* same *runtime and query complexities as $\mathcal{A}_{\xi,\chi}$ up to an additive $O(p)$ in runtime.*

Proof Let $\mathcal{A}_{\xi,\chi}$ be the randomized algorithm which takes input $\boldsymbol{x} \in \mathbb{R}^d$, random seed $\xi \sim \mathcal{D}_\xi$, and χ where $\chi = (\chi', \nu) \sim \mathcal{D}^{\boldsymbol{x}}_{\chi}$ is the concatenation of an independently drawn seed $\chi' \sim \mathcal{D}^{\boldsymbol{x}}_{\chi'}$ and seed $\nu \sim \mathcal{D}_\nu := \mathsf{Unif}^p(-t, t)$, where we use Unif^p to denote the distribution of a p-dimensional independent uniform random vector, and t is some parameter to be specified later in this proof.

For any realization $s, c, \boldsymbol{e}$ of $\xi, \chi', \boldsymbol{\nu}$, let $\mathcal{A}'_{\xi=s,\chi'=(c,\boldsymbol{e})}(x) = \mathcal{A}_{\xi=s,\chi=c} + \boldsymbol{e}$. That is, on a given input $\boldsymbol{x}$, $\mathcal{A}'_{\xi,\chi'}(\boldsymbol{x})$ has the same distribution of a random variable which is equal to $\mathcal{A}_{\xi,\chi}(\boldsymbol{x})$ plus some independent $\mathsf{Unif}(-t, t)$ random noise in each coordinate.

First, we construct a smoothing for $\mathcal{A}'_{\xi,\chi}$. Let $\bar{\mathcal{A}}_\chi$ be the randomized algorithm which takes input $\boldsymbol{x} \in \mathbb{R}^d$ and a random seed $\chi \sim \mathcal{D}^{\boldsymbol{x}}_{\chi}$. For any realization $(c, \boldsymbol{e})$ of $\chi' = (\chi, \boldsymbol{\nu})$, let $\bar{\mathcal{A}}_{\chi'}(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\nu}$. Now, using that $\mathcal{A}_{\xi,\chi}$ is an (η', δ)-approximation of f, we can conclude that

$$\mathbb{P}_{\xi\sim\mathcal{D}_\xi, \chi\sim\mathcal{D}^{\boldsymbol{x}}_{\chi}}\left(\|\mathcal{A}_{\xi,\chi}(\boldsymbol{x}) - f(\boldsymbol{x})\|_\infty \le \eta'\right) \ge 1 - \delta, \tag{1}$$

and hence

$$\mathbb{P}_{\xi\sim\mathcal{D}_\xi, \chi'\sim\mathcal{D}^{\boldsymbol{x}}_{\chi'}}\left(\|\mathcal{A}'_{\xi,\chi'}(\boldsymbol{x}) - f(\boldsymbol{x})\|_\infty \le (\eta' + t)\right) \ge 1 - \delta. \tag{2}$$

Thus, $\mathcal{A}'_{\xi,\chi'}$ is an $(\eta' + t, \delta)$-approximation of f. In particular, setting $t = \tau = \max\left(\frac{\eta'}{2}, \frac{\eta' p}{2\epsilon}\right)$ and recalliing $\eta' = \min\left(\frac{\eta}{2}, \frac{\eta\epsilon}{p}\right)$, we have

$$\eta' + t \le \eta' \max\left(\frac{3}{2}, \frac{p}{\epsilon}\right) \le \eta$$

and hence $\mathcal{A}'_{\xi,\chi'}$ is an (η, δ)-approximation of f. To bound the total variation distance, note that by (24), we have that with probability $1 - \delta$, $d_{TV}\left(p_{A_{\xi=s,\chi}(x)}, p_{\bar{A}_\chi(\boldsymbol{x})}\right)$ is simply the total variation

distance between q, q' where q is the distribution of a $\mathsf{Unif}^p[-t,t]$ random variable and q' is the distribution of a random variable whose i-th entry is distributed as $\mathsf{Unif}[-\alpha_i, \alpha_i]$ for some $|\alpha_i| \leq \eta'$. Now, for $t \geq \eta'/2$, we have

$$d_{TV}\left(q, q'\right) = 1 - \prod_{i \in [p]} \left(\frac{\max(0, 2t - |\alpha_i|)}{2t}\right) \leq 1 - \left(1 - \frac{\eta'}{2t}\right)^p \leq \frac{\eta' p}{2t},$$

where the final step used Bernoulli's inequality. Thus, setting $t = \tau = \max\left(\frac{\eta'}{2}, \frac{\eta' p}{2\epsilon}\right)$, we have that with probability $1 - \delta$,

$$\mathbb{P}_{s \sim \mathcal{D}_\chi}\left\{d_{TV}\left(p_{A'_{\xi=s,\chi'}(x)}, p_{\bar{A}_\chi(\boldsymbol{x})}\right) \leq \epsilon\right\} \geq 1 - \delta.$$

For the second guarantee, note that $\mathcal{A}'_{\xi,\chi'}$ has the same runtime and query complexities as $\mathcal{A}_{\xi,\chi}$ up to an additive $O(p)$ increase in the runtime due to the cost of adding the p-dimensional uniform random noise induced by $\boldsymbol{\nu}$. ■

We discuss implications to numerical stability further in Appendix A.

Theorem 12 show how to convert standard high-accuracy structured optimization algorithms into *pseudo-independent* high-accuracy structured optimization algorithms (Section 2) by adding noise to the output. Next, we will prove these pseudo-independent versions are amenable to *reusing* samples across multiple iterations of an outer-solver (Figure 1.) We briefly remark that this technique of adding noise to an algorithm's output also appears in differential privacy (Asi and Duchi, 2020; Ghazi et al., 2022) and in the design and analysis of algorithms that are robust to adaptive adversaries (Chen et al., 2022; Lee and Sidford, 2015; Beimel et al., 2022; van den Brand et al., 2022).

Analyzing repeated compositions of pseudoindependent functions Once we know that $\mathcal{A}'_{\xi,\chi'}$ in Meta-Algorithm 1 is *pseudo-independent* of ξ for appropriately chosen τ, we move on to proving Theorem 6.

Recall that, as outlined in Section 2, our goal will be to show that if $\mathcal{A}_{\xi,\chi'}$ in Meta-Algorithm 1 is (ϵ, δ)-pseudo-independent of ξ, then the distribution over outputs of $\texttt{Outer}(X; \texttt{Noisy}, \tau)$ and $\texttt{Outer}(X; \texttt{Reuse}, \tau)$ are close in TV. To prove this, first observe that $\texttt{NoisyLoop}_\tau$ repeatedly composes ζ with $\mathcal{A}'_{\xi,\chi'}$. Definition 13 introduces notation for these compositions.

In what follows, Eq. (3) represents an algorithm that repeatedly applies T iterations, where in each iteration ξ, χ are fresh random variables drawn from their respective distributions (as in $\texttt{NoisyLoop}_\tau$). Eq. (4) is the analogous algorithm where in the i-th iteration a fresh χ' is drawn from $\mathcal{D}^{\boldsymbol{u}}_{\chi'}$, but $\xi = s$ is *reused* across iterations (as in $\texttt{ReuseLoop}_\tau$.) Eq. (5) is the analogous expression to (3) with the randomized algorithm $\bar{\mathcal{A}}_{\chi'}$, which takes only *one* source of randomness, χ'.

Definition 13 (Composition of randomized algorithms) *Let $\mathcal{A}'_{\xi,\chi'}$ be a randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and random seeds $\xi \sim \mathcal{D}_\xi, \chi' \sim \mathcal{D}^x_{\chi'}$ and outputs a point in $\mathbb{R}^p$. Let $\bar{\mathcal{A}}_{\chi'}$ be a randomized algorithm that takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and one random seed $\chi' \sim \mathcal{D}^x_{\chi'}$ and outputs a point in $\mathbb{R}^p$. Let $\zeta : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ be a deterministic function. For $T \geq 1$, let:*

$$\begin{aligned} \Phi^1_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_{\chi'}) &:= \zeta\left(\boldsymbol{u}, \mathcal{A}'_{\xi,\chi'}(\boldsymbol{u})\right), \\ \Phi^{T+1}_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_{\chi'}) &:= \zeta\left(\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_{\chi'}),\ \mathcal{A}'_{\xi,\chi'}\left(\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_{\chi'})\right)\right) \end{aligned} \tag{3}$$

$$\begin{aligned}\Phi^1_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'}) &:= \zeta\big(\boldsymbol{u}, \mathcal{A}'_{\xi=s,\chi'}(\boldsymbol{u})\big),\\ \Phi^{T+1}_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'}) &:= \zeta\Big(\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'}),\ \mathcal{A}'_{\xi=s,\chi'}\big(\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'})\big)\Big)\end{aligned} \tag{4}$$

$$\begin{aligned}H^1_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_{\chi'}) &:= \zeta\big(\bar{\mathcal{A}}_{\chi'}(\boldsymbol{u})\big),\\ H^{T+1}_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_{\chi'}) &:= \zeta\Big(\Phi^T_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_{\chi'}),\ \bar{\mathcal{A}}_{\chi'}\big(\Phi^T_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_{\chi'})\big)\Big)\end{aligned} \tag{5}$$

In Appendix B, we show the following general theorem about pseudoindependence and repeated composition.

Lemma 14 *Let $\mathcal{A}'_{\xi,\chi'}$ be randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and two random seeds $\xi \sim \mathcal{D}_\xi, \chi' \sim \mathcal{D}^{\boldsymbol{u}}_{\chi'}$ and is (ϵ, δ)-pseudo-independent of ξ. Then,*

$$d_{TV}\left(p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'})}, p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_{\chi'})}\right) \le 2T(\delta + \epsilon)\,.$$

Finally, to prove Theorem 6, we can combine the previous results.

Theorem 6 *Suppose $\texttt{Outer}(X; \texttt{Standard})$ is (η, β)-robust with respect to f^{sub} and $\delta \in (0, 1)$. Let $\eta' := \min(\eta/2, \eta\delta)$. Suppose $\mathcal{A}^{\mathrm{sub}}_{\xi,\chi}$ is an (η', δ)-approximation of f^{sub}. Then we have that wp. $1 - 5n^2_{\mathrm{outer}}\delta$, $\texttt{Outer}(X; \texttt{Reuse}, \eta'/(2\delta))$ outputs a β-accurate solution to X.*

Proof For notational convenience, set $\epsilon = \delta$ and $\tau = \eta'/(2\epsilon)$. Notice that

$$\texttt{Outer}(X; \texttt{Noisy}, \tau) \overset{\mathcal{D}}{=} \sum_{t \in [n_{\mathrm{outer}}]} w(t) \cdot \Phi^t_{\mathcal{A}'}(\boldsymbol{u}_0; \mathcal{D}_\xi, \mathsf{D}_{\chi'}), \tag{6}$$

$$\texttt{Outer}(X; \texttt{Reuse}, \tau) \overset{\mathcal{D}}{=} \sum_{t \in [n_{\mathrm{outer}}]} w(t) \cdot \Phi^t_{\mathcal{A}'}(\boldsymbol{u}_0; s_1, \mathsf{D}_{\chi'}), \tag{7}$$

where $s_1 \sim \mathcal{D}_\xi$. Since $\tau \le \eta/2$ and $\eta' \le \eta/2$, by Lemma 3, (6) is a β-accurate solution to X wp. $1 - n_{\mathrm{outer}}\delta$.

Also, by Theorem 12, $\mathcal{A}'^{\mathrm{sub}}_{\xi,\chi'}$ is (ϵ, δ)-pseudo-independent of ξ. Thus, by Lemma 14, the TV distance between the t-th summand of (6) and the t-th summand of (7) is bounded by $2t(\delta + \epsilon)$.

As (6) is β-accurate wp. $1 - n_{\mathrm{outer}}\delta$, we can apply Fact 7 and take union bound over all $t \in [n_{\mathrm{outer}}]$ to conclude that wp. $1 - 2n^2_{\mathrm{outer}}(\delta + \epsilon) - n_{\mathrm{outer}}\delta \ge 1 - 5n^2_{\mathrm{outer}}\delta$, (7) is also β-accurate. ■

3. Application: Finite-sum minimization

In this section, we apply our sample reuse framework to obtain improved full batch versus sample query trade-offs for finite sum minimization (FSM). As a special case, we discuss implications for generalized linear models.

We focus on this setting for FSM due to its simplicity as it is perhaps the most illustrative and foundational setting. We consider a more general variant of FSM in Appendix 6, where we obtain rates dependent on the (potentially non-uniform) smoothness L_i of each f_i.

FSM arises in many machine learning settings, such as *empirical risk minimization*, where each f_i might be a loss function for a sampled data point $i \in [n]$ and the goal is to minimize the average

loss across all the data. In this context, a query $\boldsymbol{x}$ to a gradient oracle for F corresponds to making a pass over the *full batch* of data $i \in [n]$ to compute the gradient at $\boldsymbol{x}$. Meanwhile, querying a component oracle with i only requires accessing information pertaining to the i-th data point.

Below, we formally define the problem and the batch-sample oracle model and provide a brief sketch of how we obtain our result in Table 2 for FSM (Section 3).

Definition 15 (FSM problem) *In the* FSM problem *we are given $c > 1$ and $\boldsymbol{x}_0 \in \mathbb{R}^d$ and must output $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ such that $F(\hat{\boldsymbol{x}}) - \min_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x}) \leq 1/c \cdot (F(\boldsymbol{x}_0) - \min_{\boldsymbol{z}\in\mathbb{R}^d} F(\boldsymbol{z}))$ where $F(\boldsymbol{x}) := \frac{1}{n}\sum_{i\in[n]} f_i(\boldsymbol{x})$, F is μ strongly-convex, and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and convex.*

Definition 16 (Gradient oracle - FSM batch oracle) *When queried with $\boldsymbol{x} \in \mathbb{R}^d$, a* gradient oracle *for differentiable $F : \mathbb{R}^d \to \mathbb{R}$ returns $\nabla F(\boldsymbol{x}) \in \mathbb{R}^d$.*

Definition 17 (Component oracle - FSM sample oracle) *When queried with $i \in [n]$, a* component oracle *for $F(\boldsymbol{x}) = \frac{1}{n}\sum_{i\in[n]} f_i(\boldsymbol{x})$ for differentiable $f_i : \mathbb{R}^d \to \mathbb{R}$ returns a* gradient oracle *for f_i.*

The standard outer-solver and sub-solver. As discussed in Section 1.1, state-of-the-art query complexities for FSM can be achieved by with accelerated proximal point/Catalyst (APP) as an outer-solver (Frostig et al., 2015; Lin et al., 2015), ℓ_2-regularized FSM for the sub-problems, and stochastic variance-reduced gradient descent (SVRG) as a sub-solver (Johnson and Zhang, 2013). Formally, each iteration of APP approximately solves a λ-regularized *sub-problem*, as follows.

Definition 18 (FSM sub-problem) *Let F be as in Definition 15 and $\lambda \geq \mu$. For any $\boldsymbol{u} = (\boldsymbol{x}, \boldsymbol{v}) \in \mathbb{R}^d \times \mathbb{R}^d$, let $\rho = (\mu + 2\lambda)/\mu$ and $\boldsymbol{y}_{\boldsymbol{u}} := 1/(1+\rho^{-1/2})\boldsymbol{x} + \rho^{-1/2}/(1+\rho^{-1/2})\boldsymbol{v}$. We define*

$$f^{\text{sub}}(\boldsymbol{u}) = \operatorname*{argmin}_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2.$$

Moreover, we say that $\boldsymbol{a}' = (\boldsymbol{x}', \boldsymbol{v}')$ is a solution to the $(\boldsymbol{u}, \lambda, c)$-sub-problem if

$$F(\boldsymbol{x}') - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2 \leq \frac{1}{c}\left(F(\boldsymbol{y}_{\boldsymbol{u}}) - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2\right).$$

APP uses the following post-process (recall Meta-Algorithm 1). This is essentially intended to implement *acceleration* (sometimes called *momentum*) on the iterates.

Definition 19 (FSM post-process) *Let F be as in Definition 15 and $\lambda > 0$. For any $\boldsymbol{u} = (\boldsymbol{x}, \boldsymbol{v}) \in \mathbb{R}^d \times \mathbb{R}^d$ and $\boldsymbol{u}' = (\boldsymbol{x}', \boldsymbol{v}') \in \mathbb{R}^d \times \mathbb{R}^d$, let $\rho = (\mu + 2\lambda)/\mu$ and $\iota = 2/\mu + 1/\lambda$, and define*

$$\zeta(\boldsymbol{u}, \boldsymbol{u}') := (1 - \rho^{-1/2})\boldsymbol{v} + \rho^{-1/2}\left(\boldsymbol{y}_{\boldsymbol{x},\boldsymbol{v}} - \iota\lambda(\boldsymbol{y}_{\boldsymbol{x},\boldsymbol{v}} - \boldsymbol{x}')\right).$$

Now, APP is known to solve the original problem, given approximate solutions to these sub-problems, in the following sense.

Theorem 20 (APP, Theorem 1.1 of Frostig et al. (2015), restated - FSM outer-solver) *Let F be a μ-strongly-convex function and $\lambda \geq \mu$. There is a $c' = \text{poly}(\lambda, \mu, c, d)$ such that the following holds. Suppose that in each iteration $t \in [n_{\text{outer}}]$ of Algorithm 2, $\boldsymbol{x}_{t-1/2}$ is a solution to the $(\boldsymbol{u}_{t-1}, \lambda, c')$-sub-problem; then $\boldsymbol{u}_{n_{\text{outer}}}$ is a solution to the FSM problem.*

Algorithm 2: $\texttt{APP-SVRG}_{\lambda,\delta}(\boldsymbol{x}_0, F, c)$ Pseudocode

Input: Initial point $\boldsymbol{x}_0 \in \mathbb{R}^d$, gradient oracle and component oracle for F, error factor c
Parameters: Failure probability δ, and $\lambda \geq \mu$.
Initialize $\boldsymbol{v}_0 \leftarrow \mathbf{0} \in \mathbb{R}^d$
Initialize $\boldsymbol{u}_0 \leftarrow (\boldsymbol{x}_0, \boldsymbol{v}_0) \in \mathbb{R}^d \times \mathbb{R}^d$
Set sufficiently large $n_{\text{outer}} = \tilde{O}(\sqrt{\lambda/\mu})$
Set sufficiently large $c' = \text{poly}(\lambda, \mu, c, d)$
for *each* $t \in [n_{\text{outer}}]$ **do**
  // We draw realizations of the random seed ξ according to the following distribution.
  Draw $s_t := \{i_1, ..., i_T\} \sim \mathcal{D}_\xi$ where each $i_j \sim \mathsf{Unif}[n]$ for sufficiently large $T = \tilde{O}(L/\lambda)$
  // We include χ, a zero-bit random bit for technical consistency with Section 2
  Draw $c_t \sim \mathcal{D}_\chi := \mathsf{Unif}[0]$
  $\boldsymbol{u}_{t-1/2} = (\boldsymbol{x}_{t-1/2}, \boldsymbol{v}_{t-1/2}) \leftarrow \mathcal{A}^{\text{SVRG}|\lambda,c',\delta/n_{\text{outer}}}_{\xi=s_t,\chi=c_t}(\boldsymbol{u}_{t-1})$.
  // The post-process implements momentum, as described in Definition 19
  $\boldsymbol{u}_t = (\boldsymbol{x}_t, \boldsymbol{v}_t) \leftarrow \zeta(\boldsymbol{u}_t, \boldsymbol{u}_{t-1/2})$
end
return: $\boldsymbol{x}_{n_{\text{outer}}}$

In particular, SVRG is a commonly used *sub-solver* to solve the sub-problems required in APP (Theorem 20) efficiently. We restate this result below.

Theorem 21 (SVRG, Theorem 2.2 of Frostig et al. (2015), restated - FSM sub-problem solver) *Let F be as in Definition 15 and $\lambda \geq \mu$. There is a randomized algorithm $\mathcal{A}^{\text{SVRG}|\lambda,c,\delta}_{\xi,\chi}$ such that the following holds.*

- *The algorithm takes in the random seeds ξ, χ distributed as follows. The first random seed $\xi \sim \{i_1, ..., i_T\}$ where each $i_j \sim \mathsf{Unif}[n]$ for some $T = \tilde{O}(L/\lambda)$. The second random seed $\chi \sim \mathsf{Unif}[0]$ is a 0-bit random seed.*
- *If $\boldsymbol{x}' = \mathcal{A}^{\text{SVRG}|\lambda,c,\delta}_{\xi,\chi}(\boldsymbol{u})$, then wp. $1-\delta$ over the draw of the random seeds ξ and χ, $\boldsymbol{x}'$ is a solution to the $(\boldsymbol{u}, \lambda, c)$-sub-problem.*
- *$\mathcal{A}^{\text{SVRG}|\lambda,c,\delta}_{\xi,\chi}$ makes $\tilde{O}(1)$ batch queries and makes sample queries* only *on the indices contained within ξ.*

By combining Theorem 20 (APP) with Theorem 21 (SVRG) and taking a union bound over all n_{outer} iterations in Algorithm 2, we obtain an algorithm for solving FSM problems. This algorithm obtains a trade-off of $\tilde{O}(\sqrt{\lambda/\mu})$ full batch (gradient oracle) queries and $\tilde{O}(L/\sqrt{\lambda\mu})$ sample (component oracle) queries for any $\lambda \geq \mu$, as reported in the "Prior Work" column of Table 2.

However, using our sample reuse framework from Section 2, we can improve this trade-off!

The sample reusing sub-solver Observe that Algorithm 2 *exactly* fits into the Meta-Algorithm 1 framework from Section 2. In particular, Algorithm 2 implements $\texttt{Outer}(X, \texttt{Standard})$ where $X = (\boldsymbol{x}_0, F, c)$ is an FSM problem instance as per Definition 15. To show that we can replace the standard sub-solver with the sample-reusing sub-solver (i.e., $\texttt{Outer}(X, \texttt{Reuse})$), we need to invoke Theorem 6. To do so, we need to show (1) that APP satisfies robustness with respect to f^{sub}

in the sense of Definition 2 and (2) that SVRG can compute an approximation in the ℓ_∞ norm, as per Definition 1. We prove this below

Lemma 22 (FSM outer-solver robustness) *Let F be as in Definition 15 and $\lambda > 0$. There is a $c' = \mathrm{poly}(\lambda, \mu, c, d)$ such that the following holds. Suppose that in each iteration $t \in [n_{\mathrm{outer}}]$ of the algorithm,*

$$\left\|\boldsymbol{u}_{t-1/2} - f^{\mathrm{sub}}(\boldsymbol{y}_{\boldsymbol{u}_{t-1}})\right\|_\infty^2 \leq \frac{1}{c'}\left(F(\boldsymbol{y}_{\boldsymbol{u}_{t-1}}) - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}_{t-1}}\right\|_2^2\right),$$

Then $\boldsymbol{u}_{n_{\mathrm{outer}}}$ is a solution to the FSM problem.

Proof We prove the lemma by invoking Theorem 20. That is, it suffices to show that for any c, λ, there exists a $c' = \mathrm{poly}(\lambda, \mu, c, d)$ such that whenever $\boldsymbol{u}'$ satisfies

$$\left\|\boldsymbol{u}' - f^{\mathrm{sub}}(\boldsymbol{y}_{\boldsymbol{u}})\right\|_\infty^2 \leq \frac{1}{c'}\left(F(\boldsymbol{y}_{\boldsymbol{u}}) - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2\right), \tag{8}$$

we have that

$$F(\boldsymbol{u}') - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2 \leq \frac{1}{c}\cdot\left(F(\boldsymbol{y}_{\boldsymbol{u}}) - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2\right). \tag{9}$$

To prove this we will use the smoothness of the function $F_{\lambda,\boldsymbol{u}}$, which is defined as follows:

$$F_{\lambda,\boldsymbol{u}}(\tilde{\boldsymbol{x}}) := F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2.$$

First, we have that (8) implies

$$\left\|\boldsymbol{u}' - f^{\mathrm{sub}}(\boldsymbol{y}_{\boldsymbol{u}})\right\|_2^2 \leq d\left\|\boldsymbol{u}' - f^{\mathrm{sub}}(\boldsymbol{y}_{\boldsymbol{u}})\right\|_\infty^2 \leq \frac{2d}{c'}\left(F(\boldsymbol{y}_{\boldsymbol{u}}) - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2\right).$$

Thus, by L-smoothness, whenever $c' > cd(L + \lambda)$, (9) holds. The result follows by Theorem 20. ■

Lemma 23 (FSM sub-problem solver high-precision) *Let F be as in Definition 15. There is a randomized algorithm $\mathcal{A}_{\xi,\chi}^{\mathrm{SVRG-HP}|\lambda,c,\delta}$ such that the following holds.*

- *The algorithm takes in the random seeds ξ, χ distributed as follows. The first random seed $\xi \sim \{i_1, ..., i_T\}$ where each $i_j \sim \mathsf{Unif}[n]$ and $T = \tilde{O}(L/\lambda)$. The second random seed $\chi \sim \mathsf{Unif}[0]$ is a 0-bit random seed.*
- *If $\boldsymbol{x}' = \mathcal{A}_{\xi,\chi}^{\mathrm{SVRG-HP}|\lambda,c,\delta}(\boldsymbol{u})$, then wp. $1 - \delta$ over the draw of the random seeds ξ and χ,*

$$\left\|\boldsymbol{x}' - f^{\mathrm{sub}}(\boldsymbol{y}_{\boldsymbol{u}})\right\|_\infty^2 \leq \frac{1}{c}\cdot\left(F(\boldsymbol{y}_{\boldsymbol{u}}) - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y}_{\boldsymbol{u}}\right\|_2^2\right).$$

- $\mathcal{A}_{\xi,\chi}^{\mathrm{SVRG-HP}|\lambda,c,\delta}$ *makes $\tilde{O}(1)$ batch queries and makes sample queries* only *on indices in ξ.*

Proof We prove the lemma by invoking Theorem 21. It is enough to show that there is a $c' = \mathrm{poly}(\lambda, \mu, c, d)$ such that whenever $\boldsymbol{x}'$ satisfies

$$F(\boldsymbol{x}') - \min_{\tilde{\boldsymbol{x}} \in \mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2} \|\tilde{\boldsymbol{x}} - \boldsymbol{y_u}\|_2^2 \leq \frac{1}{c'} \cdot \left(F(\boldsymbol{y_u}) - \min_{\tilde{\boldsymbol{x}} \in \mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2} \|\tilde{\boldsymbol{x}} - \boldsymbol{y_u}\|_2^2 \right),$$

we also have that

$$\left\| \boldsymbol{x}' - f^{\mathrm{sub}}(\boldsymbol{y_u}) \right\|_\infty^2 \leq \frac{1}{c} \cdot \left(F(\boldsymbol{y_u}) - \min_{\tilde{\boldsymbol{x}} \in \mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2} \|\tilde{\boldsymbol{x}} - \boldsymbol{y_u}\|_2^2 \right),$$

We will use the strong convexity of the function $F_{\lambda, \boldsymbol{u}}$, which is defined as follows:

$$F_{\lambda, \boldsymbol{x}}(\tilde{\boldsymbol{x}}) := F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2} \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 .$$

Now, by μ-strong convexity,

$$\left\| \boldsymbol{x}' - f^{\mathrm{sub}}(\boldsymbol{y_u}) \right\|_\infty^2 \leq \left\| \boldsymbol{x}' - f^{\mathrm{sub}}(\boldsymbol{y_u}) \right\|_2^2 \leq \frac{\mu}{2c} \cdot \left(F(\boldsymbol{y_u}) - \min_{\tilde{\boldsymbol{x}} \in \mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2} \|\tilde{\boldsymbol{x}} - \boldsymbol{y_u}\|_2^2 \right),$$

thus it suffices to set $c' = c\mu/2$. ■

Algorithm 3: `APP-SVRG-Reuse`$_{\lambda,\delta}(\boldsymbol{z}_0, F, c)$ Pseudocode

Input: Initial point $\boldsymbol{z}_0 \in \mathbb{R}^d$, gradient oracle and component oracle for F, error factor c
Parameters: Failure probability δ, and $\lambda \geq \mu$.
Initialize $\boldsymbol{u}_0 \leftarrow (\boldsymbol{x}_0, \boldsymbol{0}) \in \mathbb{R}^d \times \mathbb{R}^d$
Set sufficiently large $n_{\mathrm{outer}} = \tilde{O}(\sqrt{\lambda/\mu})$
// We draw a realization of the random seed ξ according to the following distribution.
Draw $s_1 := \{i_1, ..., i_T\} \sim \mathcal{D}_\xi$ where each $i_j \sim \mathsf{Unif}[n]$ for sufficiently large $T = \tilde{O}(L/\lambda)$
for *each* $t \in [n_{\mathrm{outer}}]$ **do**
  // Draw c_t' from the noisy distribution $\mathcal{D}_{\chi'}$ as in Meta-Algorithm 1.
  Draw $c_t' \sim \mathcal{D}_{\chi'}$
  // Implement the noisy analog of $\mathcal{A}^{\mathrm{SVRG-HP}|\lambda,\mathrm{poly}(\lambda,\mu,c),\delta/(5n_{\mathrm{outer}}^2)}$ as in Meta-Algorithm 1.
  $\boldsymbol{u}_{t-1/2} = (\boldsymbol{x}_{t-1/2}, \boldsymbol{v}_{t-1/2}) \leftarrow \mathcal{A}'^{\mathrm{SVRG-HP}|\lambda,\mathrm{poly}(\lambda,\mu,c),\delta/(5n_{\mathrm{outer}}^2)}_{\xi=s_1,\chi'=c_t}(\boldsymbol{u}_{t-1})$.
  // The post-process implements momentum, as described in Definition 19
  $\boldsymbol{u}_t = (\boldsymbol{x}_t, \boldsymbol{v}_t) \leftarrow \zeta(\boldsymbol{u}_t, \boldsymbol{u}_{t-1/2})$
end
return: $\boldsymbol{x}_{n_{\mathrm{outer}}}$

By combining Lemma 23 with Lemma 22 and applying Theorem 6, we immediately obtain our improved trade-off.

Theorem 24 (FSM improvement) *There is an algorithm (Algorithm 3) that for F as in Definition 15, $\lambda \geq \mu$, and $\delta \in (0, 1)$, makes $\tilde{O}(\sqrt{\lambda/\mu})$-batch queries and $\tilde{O}(L/\lambda)$-sample queries and solves the FSM problem wp. $1 - \delta$.*

Proof We apply Theorem 6 using $\mathcal{A}^{\mathrm{SVRG-HP}|c',\lambda,\delta/(5n_{\mathrm{outer}}^2)}_{\xi,\chi}$ as the sub-solver to solve the $(\boldsymbol{u}_{t-1}, \lambda, c)$ sub-problem in each iteration, where c' is as required by Lemma 22. The lower failure probability $\delta/(5n_{\mathrm{outer}}^2)$ is used in order to counter the blowup in failure probability in Theorem 6. ■

3.1. Applications to regression with generalized linear models (GLM)

Regression with generalized linear models is a special case of FSM where we have n data vectors $\{\boldsymbol{a}_i\}_{i=1}^n \in \mathbb{R}^d$ with n corresponding, *explicitly known*, labels $\{b_i\}_{i=1}^n \in \mathbb{R}$, and $f_i = \phi_i(\boldsymbol{a}_i^\top \boldsymbol{x})$ for some explicit function $\phi_i(z)$ (e.g., for least-squares regression, $\phi_i(z) = 1/2(z-b_i)^2$ and for logistic regression, $\phi_i(z) = \log(1+\exp(-zb_i))$.

Because the gradient mapping $g : z \mapsto \nabla\phi_i(z)$ is known explicitly, component-oracles for f_i are easily implementable: we can simply query $i \sim \boldsymbol{p}$, lookup $\boldsymbol{a}_i$, and return the function $\nabla\phi_i(\boldsymbol{a}_i^\top \boldsymbol{x}) = \boldsymbol{a}_i \cdot g(\boldsymbol{a}_i^\top \boldsymbol{x})$ explicitly for *any* future point $\boldsymbol{x}$.

So, in these settings, our improved sample query complexity corresponds to an improved trade-off between the number of full passes over $\{\boldsymbol{a}_i\}_{i=1}^m$ and the number of random samples $\boldsymbol{a}_{i\sim\mathsf{Unif}[n]}$ required to train a GLM. The implications of this result are two-fold:

- First, this improvement means that our results prove that problems such as fitting a generalized linear model can be solved with *less information* than was known previously.
- Second, our results show that one can *reuse* the same samples $\boldsymbol{a}_{i\sim\mathsf{Unif}[n]}$ across all iterations of the outer solver. In distributed memory settings or in settings where caching significantly affects computational costs, this ability to reuse the same cached samples might be beneficial in reducing memory retrieval or communication costs.

3.2. Extension to non-uniform smoothness

Our method can also be extended to non-uniformly smooth f_i, where we relax the assumption that each f_i is L-smooth in Definition 15 to assume that each f_i is L_i-smooth. In this setting, can develop improved trade-offs that depend on the distribution of the L_i's rather than the worst-case smoothness $\max_i L_i$ by instantiating APP with the primal-dual FSM sub-problem solver of Jin et al. (2022). We defer a discussion of this setting to Section 6.

A note on pseudocode in the remainder of this paper. We include the pseudocode Algorithm 2 and Algorithm 3 in this section in order to provide a clear example of a concrete instantiation of Meta-Algorithm 1 for FSM.

However, for the sake of brevity, in later appendix sections, we will not rewrite the instantiation of Meta-Algorithm 1 for each application. Instead, we will simply define the initializations; setting for n_{outer}; post-processing functions; sub-solvers, distributions, and target functions f^{sub}—as this fully characterizes the instantiations of Meta-Algorithm 1 for each application. Additionally, we include thorough references to related work which contains application-specific pseudocodes for the outer-solvers and sub-solevrs related to each application.

An exception is in the case of discounted MDPs (Section 4.2), where we *will* include pseudocode. This is because our outer-solver for DMDPs is a new contribution of our work, so we feel it is helpful to include the full pseudocode for completeness.

4. Application: Infinite-horizon Markov Decision Processes (MDPs)

In this section, we discuss how our framework can be applied to infinite-horizon Markov Decision Processes (MDPs). We primarily focus on the discounted case (DMDPs) in Section 4.2 and then in Section 4.3 extend our results to the average-reward case (AMDPs) using a standard reduction due to Jin and Sidford (2021). We begin with preliminaries in Section 4.1.

4.1. Preliminaries

We denote an MDP by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \boldsymbol{P})$ where $\mathcal{S}$ denotes a finite state space, $\mathcal{A}$ denotes the total set of all state-action pairs. Concretely, we use $\mathcal{A}_s$ to denote the set of available actions in state $s \in \mathcal{S}$ and define $\mathcal{A} = \{(s, a) : s \in \mathcal{S}, a \in \mathcal{A}_s\}$ as well as $\mathcal{A}_{\text{tot}} := |\mathcal{A}|$. The state-action-state transition matrix is given by $\boldsymbol{P} \in \mathbb{R}^{\mathcal{A} \times \mathcal{S}}$. We use $\boldsymbol{p}(s, a) \in \Delta^{\mathcal{S}}$ to denote the (s, a)-th row of $\boldsymbol{P}$. A policy π is a mapping from states to actions, i.e., $\pi : s \mapsto \pi(s) \in \mathcal{A}_s$. We use Π to denote the set of all possible policies.

Throughout this section, for vectors $\boldsymbol{a}, \boldsymbol{b}$, we use $\boldsymbol{a} \geq \boldsymbol{b}$ to mean entrywise inequality (and use $\leq, >, <$ analogously.) For $\mathcal{A}_{\text{tot}}$-dimensional vectors, say $\boldsymbol{r} \in \mathbb{R}^{\mathcal{A}}_{\text{tot}}$, we use the notation $\boldsymbol{r}(s, a)$ to denote the (s, a)-th entry of $\boldsymbol{r}$. We assume rewards are known a-priori, as in prior works in this setting (see, e.g., Jin et al. (2024) and references therein.)

For any policy π, we use $\boldsymbol{r}^\pi \in \mathbb{R}^{\mathcal{S}}_{\geq 0}$ to denote the $|\mathcal{S}|$-dimensional vector with s-th entry given by $\boldsymbol{r}^\pi(s, \pi(s))$. Likewise, we use $\boldsymbol{P}^\pi \in \mathbb{R}^{\mathcal{S} \times \mathcal{S}}$ to denote the sub-matrix of $\boldsymbol{P}$ where the s-th row of $\boldsymbol{P}^\pi$ is $\boldsymbol{p}(s, \pi(s))$.

An agent interacts with the MDP in *timestep* as follows. At each timestep $t \geq 0$, an agent begins in state s_t, chooses an available action $a_t \in \mathcal{A}_{s_t}$, and collects a (finite) reward $\boldsymbol{r}(s_t, a_t)$. The agent then (stochastically) transitions to the next state s_{t+1} where $s_{t+1} \sim \boldsymbol{p}(s, a)$. The agent's goal is to compute a *policy* for selecting actions in each state that maximizes the agent's infinite-horizon expected utility over the set of all policies, denoted Π.

This objective is often formalized in two settings: the discounted setting (DMDP) and the average-reward setting (AMDP), which we will discuss in Section 4.2 and Section 4.3 respectively. However, we will first define the full batch and sample oracle access for $\mathcal{M}$.

Definition 25 (Matrix-vector oracle - DMDP/AMDP batch oracle) *When queried with $\boldsymbol{x} \in \mathbb{R}^{\mathcal{S}}$, a* matrix-vector oracle *for $\boldsymbol{P}$ returns $\boldsymbol{P}\boldsymbol{x}$.*

Definition 26 (Simulator oracle - DMDP/AMDP sample oracle) *When queried with $(s, a) \in \mathcal{A}$, a* simulator oracle *returns a sample $s' \sim \boldsymbol{p}(s, a)$.*

The oracle described in Definition 26 is often called a *generative model* in the reinforcement learning theory literature (see, for example, Kearns and Singh (1998)).

4.2. Discounted MDP

We begin by describing MDPs in the *discounted setting*.

Discounted MDP (DMDP). In a DMDP, the objective is to compute an ϵ-optimal policy for maximizing the infinite-horizon γ-discounted reward for some known constant $\gamma \in (0, 1)$.

Definition 27 (Discounted MDP (DMDP)) *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \boldsymbol{P})$ be an MDP, $\gamma \in (0, 1)$, and $\boldsymbol{r} \in [0, 1]^{\mathcal{A}}$. We denote the associated γ-discounted-MDP by $\mathcal{M}_{\gamma, \boldsymbol{r}} = (\mathcal{M}; \gamma, \boldsymbol{r})$.*

In a DMDP, the value of a policy π is defined as follows.

Definition 28 (DMDP policy value) *Let $\mathcal{M}_{\gamma,\boldsymbol{r}}$ be a DMDP and π be a policy. The value of policy π, denoted $\boldsymbol{v}^{\pi}_{\gamma,\boldsymbol{r}}$, is defined as*

$$\boldsymbol{v}^{\pi}_{\gamma,\boldsymbol{r}}(s) := \mathbb{E}\left[\sum_{t\geq 0}\gamma^t \boldsymbol{r}(s_t,\pi(s_t))|s_0 = s\right].$$

We can alternatively define the value of a policy in terms of the Bellman operator.

Definition 29 (Bellman operator) *Let $\mathcal{M}_{\gamma,\boldsymbol{r}}$ be a DMDP. Given a policy π, and vector $\boldsymbol{v} \in \mathbb{R}^{\mathcal{S}}$ we define the* Bellman operator *associated with π as $\mathcal{T}^{\pi}_{\gamma,\boldsymbol{r}}[\boldsymbol{v}] \in \mathbb{R}^{\mathcal{S}}$ as*

$$\mathcal{T}^{\pi}_{\gamma,\boldsymbol{r}}[\boldsymbol{v}](s) := \boldsymbol{r}(s,\pi(s)) + \gamma \boldsymbol{p}(s,\pi(s))^{\top}(s,\pi(s)).$$

The value of policy π is the unique vector such that $\boldsymbol{v}^{\pi}_{\gamma,\boldsymbol{r}} = \mathcal{T}^{\pi}_{\gamma,\boldsymbol{r}}[\boldsymbol{v}^{\pi}_{\gamma,\boldsymbol{r}}]$. We also define the Bellman operator $\mathcal{T}_{\gamma,\boldsymbol{r}}$ to be the operator that maps $\boldsymbol{v} \mapsto \mathcal{T}_{\gamma,\boldsymbol{r}}[\boldsymbol{v}] \in \mathbb{R}^{\mathcal{S}}$ where

$$\mathcal{T}_{\gamma,\boldsymbol{r}}[\boldsymbol{v}](s) := \max_{a\in\mathcal{A}_s} \boldsymbol{r}(s,a) + \gamma \boldsymbol{p}(s,a)^{\top}\boldsymbol{v}.$$

Fact 30 (Bellman optimality conditions) *The value of the optimal policy $\pi^{\star}_{\gamma,\boldsymbol{r}}$ for $\mathcal{M}_{\gamma,\boldsymbol{r}}$ satisfies*

$$\boldsymbol{v}^{\pi^{\star}_{\gamma,\boldsymbol{r}}}_{\gamma,\boldsymbol{r}}(s) = \max_{\pi\in\Pi}\mathbb{E}\left[\sum_{t\geq 0}\gamma^t \boldsymbol{r}(s_t,\pi(s_t))|s_0 = s\right],$$

and $\boldsymbol{v}^{\pi^{\star}_{\gamma,\boldsymbol{r}}}_{\gamma,\boldsymbol{r}}$ is the unique vector such that $\boldsymbol{v}^{\pi^{\star}_{\gamma,\boldsymbol{r}}}_{\gamma,\boldsymbol{r}} = \mathcal{T}_{\gamma,\boldsymbol{r}}[\boldsymbol{v}^{\pi^{\star}_{\gamma,\boldsymbol{r}}}_{\gamma,\boldsymbol{r}}]$. We use the shorthand $\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} = \boldsymbol{v}^{\pi^{\star}_{\gamma,\boldsymbol{r}}}_{\gamma,\boldsymbol{r}}$ and refer to it as the optimal value vector *for $\mathcal{M}_{\gamma,r}$.*

Equipped with these definitions of values in DMDPs, we define the DMDP problem as follows.

Definition 31 (DMDP problem) *Let $\mathcal{M}_{\gamma,\boldsymbol{r}}$ be a γ-discounted MDP and $\epsilon \in (0,(1-\gamma)^{-1}]$ be given. We must compute a value $\boldsymbol{v}$ and a policy π such that*

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v} \leq \epsilon\mathbf{1}, \quad \text{and} \quad \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \epsilon\mathbf{1} \leq \boldsymbol{v}^{\pi}_{\gamma,\boldsymbol{r}} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}.$$

Such a policy π and value $\boldsymbol{v}$ is called an ϵ-optimal policy and ϵ-optimal value for $\mathcal{M}_{\gamma,\boldsymbol{r}}$.

The standard outer-solver and sub-solver. Now, we will show how to reduce the DMDP problem for a discount factor $\gamma > 0$ to solving a sequence of sub-problems. In this case, the sub-problem will essentially require solving a DMDP with a *smaller* discount factor γ'. Note that in the sub-problems, we relax the requirement from $\boldsymbol{r} \in [0,1]^{\mathcal{A}}$ to simply $\boldsymbol{r} \in \mathbb{R}^{\mathcal{A}}_{\geq 0}$.

Definition 32 (DMDP sub-problem) *Let $\mathcal{M} = (\mathcal{S},\mathcal{A},\boldsymbol{P})$ be an MDP and $\boldsymbol{r} \in \mathbb{R}^{\mathcal{A}}_{\geq 0}$ be a reward vector. In the $(\mathcal{M},\boldsymbol{r},\gamma',\boldsymbol{v},\epsilon)$-sub-problem, we are given $\gamma' > 0$, $\boldsymbol{v} \in \mathbb{R}^{\mathcal{S}}$ and $\epsilon \in (0,1/(1-\gamma')]$. We define $\boldsymbol{r}' := \boldsymbol{r} - (\gamma'-\gamma)\boldsymbol{P}\boldsymbol{v}$, and must compute a value $\boldsymbol{v}$ such that $\|\boldsymbol{v}^{\star}_{\gamma',\boldsymbol{r}'} - \boldsymbol{v}\|_{\infty} \leq \epsilon/2$.*

To enable computing approximately optimal policies in addition to approximately optimal values, we also introduce the following *policy*-sub-problem as follows.

Definition 33 (DMDP policy-sub-problem) *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \boldsymbol{P})$ be an MDP and $\boldsymbol{r} \in \mathbb{R}^{\mathcal{S}}_{\geq 0}$ be a reward vector. In the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}, \epsilon)$-policy-sub-problem, we are given $\gamma' > 0$, $\boldsymbol{v} \in \mathbb{R}^{\mathcal{S}}, \epsilon \in (0, 1/(1-\gamma')]$*

$$\boldsymbol{r}' := \boldsymbol{r} - (\gamma' - \gamma)\boldsymbol{P}\boldsymbol{v},$$

and we must compute a value $\boldsymbol{v}$ and a policy π such that $\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma', \boldsymbol{r}'} - \boldsymbol{v} \leq \epsilon \mathbf{1}$ and $\boldsymbol{v} \leq \boldsymbol{v}^{\pi}_{\gamma', \boldsymbol{r}'}$. That is, we must find an ϵ-optimal value and policy for $\mathcal{M}_{\gamma', \boldsymbol{r}'}$.

In comparison to the DMDP sub-problem (Definition 32), in the DMDP policy-sub-problem, we must not only output an ϵ-optimal value for $\mathcal{M}_{\gamma', \boldsymbol{r}'}$ but must *also* output a policy π that attains at least that value. For the DMDP problem, the outer process is defined (simply) as follows.

Definition 34 (DMDP post-process) *Fix $\epsilon \in (0, 1/(1-\gamma')]$. For any $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^{\mathcal{S}}$, we define $\zeta_{\epsilon}(\boldsymbol{v}, \boldsymbol{v}') := \boldsymbol{v}' - \epsilon \mathbf{1}$.*

The outer process is intented to adjust (shift down the entries of) $\boldsymbol{v}'$ to ensure that it is *an underestimate* of the optimal value.

We prove the following outer-solver for DMDPs, which we call the Proximal Reward Method (PRM), since it is in spirit similar to proximal point methods in convex optimization.

Theorem 35 (PRM - DMDP outer-solver) *Let $\mathcal{M}_{\gamma, \boldsymbol{r}}$ be a DMDP, $\gamma' < \gamma$, and $\epsilon \in (0, 1/(1-\gamma)]$. Let $\epsilon' = \epsilon/4 \cdot (1-\gamma)/(1-\gamma')$. Suppose that in each iteration $t \in [n_{\text{outer}}]$ of Algorithm 4, $\boldsymbol{v}_{t-1/2}$ is a solution to the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}_{t-1}, \epsilon')$-sub-problem. Suppose that $\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1}$ is a solution to the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}_{n_{\text{outer}}}, \epsilon')$-policy sub-problem. Then, $\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1}$ solves the DMDP problem.*

We prove Theorem 35 in Section 4.4, and for now, focus on its application. To do so, we need to discuss high-precision algorithms for solving the DMDP sub-problem and policy-sub-problem.

Subproblem solvers. There are many high-accuracy algorithms for solving the DMDP subproblem as well as the DMDP-policy-sub-problem. These methods can broadly be divided into interior-point methods (e.g., (Sidford et al., 2018b; Lee and Sidford, 2014; van den Brand et al., 2021; Cohen et al., 2020a; Jiang et al., 2021)), classical value iteration (VI) (Littman et al., 1995; Tseng, 1990), variance-reduced variants variants of (VI), as well as policy-based methods such as policy iteration or policy gradient and variants (see e.g., Grondman et al. (2012); Bertsekas (2011) for a survey.) These variants (Sidford et al., 2018b; Jin et al., 2024) implement an approximate version of VI by trading-off between full batch queries (matrix-vector products in $\boldsymbol{P}$) and sample queries (simulator oracle queries) to obtain faster runtimes. Since we are interested in trade-offs between full batch and sample queries, we consider the Truncated Variance-Reduced Value Iteration (TVRVI) outer-solver from Jin et al. (2024) since it achieves the best trade-off between full batch and sample queries in this setting.

Theorem 36 (TVRVI, Theorem 1.2 of Jin et al. (2024), restated - DMDP sub-problem solver) *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \boldsymbol{P})$ be an MDP, $\boldsymbol{r} \in \mathbb{R}^{\mathcal{A}}_{\geq 0}$ and $0 < \gamma' < \gamma < 1$. There are randomized algorithms $\mathcal{A}^{\text{SVRG}|\gamma', \epsilon}_{\xi, \chi}(\boldsymbol{x})$ and $\mathcal{A}^{\text{SVRG}-\text{Policy}|\gamma', \epsilon}_{\xi, \chi}(\boldsymbol{x})$ such that the following hold true.*

- *The algorithms take in the random seeds ξ, χ distributed as follows. The first random seed $\xi \sim \{i_1, ..., i_T\}$ where each $i_j \in \mathcal{S}^{\mathcal{A}}$ and each $i_j(s,a) \sim \boldsymbol{p}(s,a)$ and $T = \tilde{O}((1-\gamma')^{-2})$. The second random seed $\chi \sim \mathsf{Unif}[0]$ is a 0-bit random seed.*

- *If $\boldsymbol{v}' = \mathcal{A}_{\xi,\chi}^{\text{TVRVI}|\gamma',\epsilon,\delta}(\boldsymbol{v})$, then wp. $1-\delta$ over the draw of the random seeds ξ and χ, $\boldsymbol{v}'$ is a solution to the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}, \epsilon)$-sub-problem.*

- *If $\boldsymbol{v}', \pi = \mathcal{A}_{\xi,\chi}^{\text{TVRVI-Policy}|\gamma',\epsilon,\delta}(\boldsymbol{v})$, then wp. $1-\delta$ over the draw of the random seeds ξ and χ, $\boldsymbol{v}', \pi$ are a solution to the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}, \epsilon)$-policy sub-problem.*

- *The algorithms make only $\tilde{O}(1)$ batch queries and make only the sample queries that are encoded in ξ.*

By instantiating the DMDP outer-solver PRM (Theorem 35) with TVRVI (Theorem 36) as the sub-solver, for $\gamma' \leq \gamma$, we obtain a full batch versus sample query trade-off of $\tilde{O}((1-\gamma')/(1-\gamma))$ full batch and $\tilde{O}((1-\gamma')^{-1}(1-\gamma)^{-1})$ sample queries per state-action pair. The pseudo-code is shown in Algorithm 4.

Algorithm 4: $\texttt{PRM-TVRVI}_{\gamma',\delta}(\mathcal{M}, \boldsymbol{r}, \epsilon, \gamma)$ Pseudocode

Input: Matrix-vector oracle for $\boldsymbol{P}$, simulator oracle for $\boldsymbol{P}$, reward vector $\boldsymbol{r} \in [0,1]^{\mathcal{S}}$, error tolerance $\epsilon \in (0, 1/(1-\gamma)]$, discount factor $\gamma \in (0,1)$

Parameters: Failure probability δ, and $0 < \gamma' < \gamma < 1$.

Initialize $\boldsymbol{v}_0 \leftarrow \boldsymbol{0} \in \mathbb{R}^{\mathcal{S}}$

Set sufficiently large $n_{\text{outer}} = \tilde{O}((1-\gamma')/(1-\gamma))$

Set $\epsilon' = \epsilon/4 \cdot (1-\gamma)/(1-\gamma')$

for *each* $t \in [n_{\text{outer}}]$ **do**

  // We draw realizations of the random seed ξ according to the following distribution.

  Draw $s_t := \{i_1, ..., i_T\} \sim \mathcal{D}_\xi$ where each $i_j \in \mathcal{S}^{\mathcal{A}}$ and each $i_j(s,a) \sim \boldsymbol{p}(s,a)$ for sufficiently large $T = \tilde{O}((1-\gamma')^{-2})$

  // We include a χ, a zero-bit random bit for technical consistency with Section 2

  Draw $c_t \sim \mathcal{D}_\chi := \mathsf{Unif}[0]$

  $\boldsymbol{v}_{t-1/2} \leftarrow \mathcal{A}_{\xi=s_t,\chi=c_t}^{\text{TVRVI}|\gamma',\epsilon',\delta/n_{\text{outer}}}(\boldsymbol{v}_{t-1})$.

  // The post-process is as described in Definition 34

  $\boldsymbol{v}_t \leftarrow \zeta_{\epsilon'}(\boldsymbol{v}_{t-1}, \boldsymbol{v}_{t-1/2})$

end

// For the final iteration, we again draw a realization of the random seed ξ according to the following distribution.

Draw $s := \{i_1, ..., i_T\} \sim \mathcal{D}_\xi$ where each $i_j \in \mathcal{S}^{\mathcal{A}}$ and each $i_j(s,a) \sim \boldsymbol{p}(s,a)$ for sufficiently large $T = \tilde{O}((1-\gamma')^{-2})$

// Again, we include a χ, a zero-bit random bit for technical consistency with Section 2

Draw $c_t \sim \mathcal{D}_\chi := \mathsf{Unif}[0]$

$\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1} \leftarrow \mathcal{A}_{\xi=s,\chi=c}^{\text{TVRVI-Policy}|\gamma',\epsilon',\delta/n_{\text{outer}}}(\boldsymbol{v}_{n_{\text{outer}}})$.

return: $\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1}$

The sample reusing sub-solver Next, we observe that Algorithm 4 *exactly* fits into the Meta-Algorithm 1 framework from Section 2. In particular, we observe that Algorithm 2 implements $\texttt{Outer}(X, \texttt{Standard})$ where $X = (\mathcal{M}, \boldsymbol{r}, \gamma)$ is an DMDP problem instance as per Definition 31. To show that we can replace the standard sub-solver with the sample-reusing sub-solver (i.e., $\texttt{Outer}(X, \texttt{Reuse})$), we need to invoke Theorem 6.

However, we can also observe that the DMDP outer-solver PRM in Theorem 36 is *by definition* robust with respect to the target function f^{sub} (defined below), in the sense of Definition 2. Further, the sub-solver TVRVI *already* guarantees ℓ_∞ accuracy guarantees in the sense of Definition 1 for the target function

$$f^{\text{sub}}(\boldsymbol{v}) := \boldsymbol{r} - \boldsymbol{v}^\star_{\gamma', (\gamma'-\gamma)\boldsymbol{P}\boldsymbol{v}}.$$

Consequently, we can *directly* apply our sample reuse framework to reuse samples across all n_{outer} iterations of the for loop in Algorithm 4 to obtain the following improved trade-off. In particular, by invoking Theorem 6, we obtain the following result.

Theorem 37 (DMDP trade-off improvement) *Let $\mathcal{M}_{\gamma,\boldsymbol{r}}$ be a DMDP and $\gamma' < \gamma$. There is an algorithm (Algorithm 5) which makes $\tilde{O}((1-\gamma')/(1-\gamma)$-batch queries and $\tilde{O}(\mathcal{A}_{\text{tot}}(1-\gamma')^{-2})$-sample queries and solves the DMDP problem wp. $1-\delta$.*

Faster algorithm for certain DMDPs Theorem 35 also yields another interesting implication regarding the runtime for solving a DMDP. In particular, Truncated Variance-Reduced Value Iteration Jin et al. (2024) is known to solve the $(\mathcal{M}, \gamma', \boldsymbol{r}', \epsilon, \delta)$-sub-problem in $\tilde{O}(\text{nnz}(\boldsymbol{P}) + \mathcal{A}_{\text{tot}}(1-\gamma')^{-2})$-time (Theorem 1.1 of Jin et al. (2024)) for $\boldsymbol{r} \in [0,1]^{\mathcal{S}}$.[3] Consequently, Theorem 36 solves the DMDP problem in

$$\tilde{O}\left(\frac{(1-\gamma')}{(1-\gamma)} \cdot (\text{nnz}(\boldsymbol{P}) + \mathcal{A}_{\text{tot}}(1-\gamma')^{-2})\right) = \tilde{O}\left(\frac{\text{nnz}(\boldsymbol{P})(1-\gamma')}{(1-\gamma)} + \frac{\mathcal{A}_{\text{tot}}}{(1-\gamma)(1-\gamma')}\right)$$

time for $\gamma' \le \gamma$. So, by selecting $1-\gamma' = \max\left(\sqrt{\mathcal{A}_{\text{tot}}/\text{nnz}(\boldsymbol{P})}, 1-\gamma\right)$ to minimize *runtime*, we obtain the following immediate corollary of Theorem 35.

Theorem 38 (Faster runtime for solving certain MDPs) *Suppose $\text{nnz}(\boldsymbol{P}) \le \mathcal{A}_{\text{tot}}(1-\gamma)^{-2}$. Then, there is an algorithm that solves the DMDP problem (Definition 31) in $\tilde{O}(\text{nnz}(\boldsymbol{P}) + \sqrt{\text{nnz}(\boldsymbol{P})\mathcal{A}_{\text{tot}}}(1-\gamma)^{-1})$-time.*

Proof If $\text{nnz}(\boldsymbol{P}) \le \mathcal{A}_{\text{tot}}(1-\gamma)^{-2}$, then taking

$$1-\gamma' = \max(\sqrt{\mathcal{A}_{\text{tot}}/\text{nnz}(\boldsymbol{P})}, (1-\gamma)) = \sqrt{\mathcal{A}_{\text{tot}}/\text{nnz}(\boldsymbol{P})},$$

in which case the runtime becomes

$$\tilde{O}\left(\text{nnz}(\boldsymbol{P}) + \sqrt{\text{nnz}(\boldsymbol{P})\mathcal{A}_{\text{tot}}}(1-\gamma)^{-1}\right).$$

■

Theorem 38 improves on Jin et al. (2024) in the regime where $\text{nnz}(\boldsymbol{P}) = o((1-\gamma)^{-2}\mathcal{A}_{\text{tot}})$ and improves upon vanilla value iteration (which runs in $\tilde{O}(\text{nnz}(\boldsymbol{P})(1-\gamma)^{-1})$ in the regime where $\text{nnz}(\boldsymbol{P}) \ge \mathcal{A}_{\text{tot}}$.

3. We use $\text{nnz}(\boldsymbol{P})$ to denote the number of nonzero entries in $\boldsymbol{P}$.

Algorithm 5: PRM-TVRVI-Reuse$_{\gamma',\delta}(\mathcal{M}, \boldsymbol{r}, \epsilon, \gamma)$ Pseudocode

Input: Matrix-vector oracle for $\boldsymbol{P}$, simulator oracle for $\boldsymbol{P}$, reward vector $\boldsymbol{r} \in [0,1]^{\mathcal{S}}$, error tolerance $\epsilon \in (0, 1/(1-\gamma)]$, discount factor $\gamma \in (0,1)$

Parameters: Failure probability δ, and $0 < \gamma' < \gamma < 1$.

Initialize $\boldsymbol{v}_0 \leftarrow \mathbf{0} \in \mathbb{R}^{\mathcal{S}}$

Set sufficiently large $n_{\text{outer}} = \tilde{O}((1-\gamma')/(1-\gamma))$

Set $\epsilon' = \epsilon/4 \cdot (1-\gamma)/(1-\gamma')$

// We draw realizations of the random seed ξ according to the following distribution.

Draw $s_1 := \{i_1, ..., i_T\} \sim \mathcal{D}_\xi$ where each $i_j \in \mathcal{S}^{\mathcal{A}}$ and each $i_j(s,a) \sim \boldsymbol{p}(s,a)$ for sufficiently large $T = \tilde{O}((1-\gamma')^{-2})$

for *each* $t \in [n_{\text{outer}} - 1]$ **do**

  // Draw c'_t from the noisy distribution $\mathcal{D}_{\chi'}$ as in Meta-Algorithm 1.

  Draw $c'_t \sim \mathcal{D}_{\chi'}$

  // Implement the noisy analog of $\mathcal{A}^{\text{TVRVI}|\gamma',\epsilon',\delta/(5n_{\text{outer}}^2)}$ as in Meta-Algorithm 1.

  $\boldsymbol{v}_{t-1/2} \leftarrow \mathcal{A}'^{\text{TVRVI}|\gamma',\epsilon',\delta/(5n_{\text{outer}}^2)}_{\xi=s_t,\chi'=c'_t}(\boldsymbol{v}_{t-1})$.

  // The post-process is as described in Definition 34

  $\boldsymbol{v}_t \leftarrow \zeta_{\epsilon'}(\boldsymbol{v}_{t-1}, \boldsymbol{v}_{t-1/2})$

end

// For the final iteration, we again draw a realization of the random seed ξ according to the following distribution.

Draw $s := \{i_1, ..., i_T\} \sim \mathcal{D}_\xi$ where each $i_j \in \mathcal{S}^{\mathcal{A}}$ and each $i_j(s,a) \sim \boldsymbol{p}(s,a)$ for sufficiently large $T = \tilde{O}((1-\gamma')^{-2})$

// Again, we include a χ, a zero-bit random bit for technical consistency with Section 2

Draw $c_t \sim \mathcal{D}_\chi := \mathsf{Unif}[0]$

$\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1} \leftarrow \mathcal{A}^{\text{TVRVI-Policy}|\gamma',\epsilon',\delta/(5n_{\text{outer}}^2)}_{\xi=s,\chi=c}(\boldsymbol{v}_{n_{\text{outer}}})$.

return: $\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1}$

4.3. Infinite-horizon Average-reward MDPs

In this section, we consider the average-reward setting, in which there is no discount factor γ.

Definition 39 (Average-reward MDP (AMDP)) *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \boldsymbol{P})$ be an MDP and $\boldsymbol{r} \in [0,1]^{\mathcal{A}}$. We denote the associated AMDP by $\mathcal{M}_{\boldsymbol{r}} = (\mathcal{M}; \boldsymbol{r})$.*

In the average-reward case, we define the value of a policy as follows.

Definition 40 (AMDP policy value) *Let $\mathcal{M}_{\boldsymbol{r}}$ be an AMDP and π be a policy. The value of policy π, denoted $\boldsymbol{v}^\pi_{\boldsymbol{r}}$, is defined as*

$$\boldsymbol{v}^\pi_{\boldsymbol{r}}(s) = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}\left[\sum_{t\geq 0} \boldsymbol{r}(s_t, \pi(s_t)) | s_0 = s\right].$$

As in the DMDP case, the goal is to find an approximately optimal policy.

Definition 41 (AMDP problem) *Let $\mathcal{M}_{\boldsymbol{r}}$ be an AMDP and $\epsilon \in (0,1)$ be given. We must compute, wp. $1-\delta$, a policy π such that $\|\boldsymbol{v}^\pi_{\boldsymbol{r}} - \boldsymbol{v}^\star_{\boldsymbol{r}}\|_\infty \leq \epsilon$. Such a policy is called ϵ-optimal for $\mathcal{M}_{\boldsymbol{r}}$.*

In order to extend our improved trade-offs for DMDPs to AMDPs, we leverage a result of Jin and Sidford (2021), which showed that the AMDP problem can be reduced to solving a DMDP with a sufficiently high discount factor. In the following, Δ denotes the probability simplex.

Definition 42 (Mixing time) *Let $\mathcal{M}_{\boldsymbol{r}}$ be an AMDP. $\mathcal{M}_{\boldsymbol{r}}$ is said to be mixing if there exists a stationary distribution $\boldsymbol{\nu} \in \Delta^{\mathcal{S}}$ such that*

$$t_{\text{mix}} := \max_{\pi \in \Pi} \operatorname*{argmin}_{t \geq 1} \sum_{s \in \mathcal{S}} \max_{\boldsymbol{q} \in \Delta^{\mathcal{S}}} \boldsymbol{p}(s, \pi(s))^\top \boldsymbol{q} - \boldsymbol{\nu} < \infty.$$

The quantity t_{mix} is called the mixing time *of $\mathcal{M}_{\boldsymbol{r}}$.*

Lemma 43 (Lemma 3 of Jin and Sidford (2021)) *Let $\mathcal{M}_{\boldsymbol{r}}$ be an AMDP. Suppose the mixing time of the $\mathcal{M}_{\boldsymbol{r}}$ is $t_{\text{mix}} < \infty$. Then, for any $\epsilon > 0$ and $\gamma \in (0, 1 - \epsilon/(9t_{\text{mix}}))$, an $\epsilon/(3(1-\gamma))$-optimal policy for the DMDP $\mathcal{M}_{\gamma, \boldsymbol{r}}$ is also an ϵ-optimal policy for the AMDP $\mathcal{M}_{\boldsymbol{r}}$.*

By combining Lemma 43 with Theorem 37, we immediately obtain the following full batch and sample query trade-off for AMDPs.

Theorem 44 (AMDP trade-off improvement) *Let $\mathcal{M}_{\boldsymbol{r}}$ be an AMDP. Suppose the mixing time of $\mathcal{M}_{\boldsymbol{r}}$ is $t_{\text{mix}} < \infty$. Then, for $\gamma' \leq 1 - \epsilon/(9t_{\text{mix}})$, there is an algorithm that solves the AMDP problem using $\tilde{O}((1-\gamma')t_{\text{mix}}/\epsilon)$ full batch queries and only $\tilde{O}(\mathcal{A}_{\text{tot}}(1-\gamma')^{-2})$ sample queries.*

4.4. Proximal Reward Method for DMDPs: Proof of Theorem 35

In this section we present the proof of Theorem 35. First, we prove a stability result regarding the optimal value of a DMDP under a reward perturbation.

Lemma 45 *Let $\boldsymbol{r}, \boldsymbol{r}' \in \mathbb{R}^{\mathcal{A}}$ such that $\boldsymbol{r}' \leq \boldsymbol{r}$, and let $\gamma > 0$. Then, for any $\boldsymbol{v} \in \mathbb{R}^{\mathcal{S}}$, we have that for all $s \in \mathcal{S}$,*

$$\mathbf{0} \leq \boldsymbol{v}^\star_{\gamma, \boldsymbol{r}} - \boldsymbol{v}^\star_{\gamma, \boldsymbol{r}'} \leq \frac{1}{1-\gamma} \cdot \left(\max_{(s,a) \in \mathcal{A}} \boldsymbol{r}(s,a) - \boldsymbol{r}'(s,a) \right) \cdot \mathbf{1}.$$

Proof By the Bellman optimality conditions (Definition 29), for each $s \in \mathcal{S}$ we have

$$\boldsymbol{v}^\star_{\gamma, \boldsymbol{r}}(s) = \max_{\pi \in \Pi} \left((\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} \boldsymbol{r}^\pi \right)(s),$$
$$\boldsymbol{v}^\star_{\gamma, \boldsymbol{r}'}(s) = \max_{\pi \in \Pi} \left((\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} \boldsymbol{r}'^\pi \right)(s).$$

Since $(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}$ is a positive matrix, we have that for each $s \in \mathcal{S}$,

$$\mathbf{0} \leq \boldsymbol{v}^\star_{\gamma, \boldsymbol{r}}(s) - \boldsymbol{v}^\star_{\gamma, \boldsymbol{r}'}(s) \leq \max_{\pi \in \Pi} \left((\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} (\boldsymbol{r}^\pi - \boldsymbol{r}'^\pi) \right)(s) \leq \frac{1}{1-\gamma} \cdot \max_{(s,a) \in \mathcal{A}} \boldsymbol{r}(s,a) - \boldsymbol{r}'(s,a).$$

■

Next, we show how to iteratively solve a sequence of γ'-discounted MDPs to solve a γ-discounted MDP. First, we prove the following lemma, which bounds the convergence rate of the scheme which solves $\mathcal{M}_{\gamma, \boldsymbol{r}}$ by iteratively solving problems in $\mathcal{M}_{\gamma', \boldsymbol{r}'}$ for a sequence of rewards $\boldsymbol{r}^{(1)}, ..., \boldsymbol{r}^{(T)}$.

Lemma 46 *Let $\mathcal{M}$ be an MDP and $\boldsymbol{r} \in \mathbb{R}^{\mathcal{A}}_{\geq 0}$ be a reward vector. Let $0 < \gamma' \leq \gamma < 1$, $\eta, \epsilon > 0$, and $T \geq 1$. Let $\boldsymbol{r}^{(0)} = \boldsymbol{r}$ and $\boldsymbol{v}^{(0)} = \mathbf{0}$. For each $t \geq 1$, define*

$$\boldsymbol{r}^{(t)} := \boldsymbol{r} - (\gamma' - \gamma)\boldsymbol{P}\boldsymbol{v}^{(t-1)}. \tag{10}$$

Suppose that for each $t \geq 1$, $\boldsymbol{v}^{(t)}$ satisfies

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma', \boldsymbol{r}^{(t)}} - \eta\mathbf{1} \leq \boldsymbol{v}^{(t)} \leq \boldsymbol{v}^{\star}_{\gamma', \boldsymbol{r}^{(t)}}.$$

Then, for any $T \geq 1$,

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \left(\left(\frac{(\gamma - \gamma')}{(1-\gamma')} \right)^{T} \max_{s \in \mathcal{S}} \left(\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v}^{(0)} \right)(s) + \frac{(1-\gamma')}{(1-\gamma)}\eta \right) \cdot \mathbf{1} \leq \boldsymbol{v}^{(T)} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}.$$

Consequently, if $\eta \leq \frac{(1-\gamma)}{2(1-\gamma')}\epsilon$ and $T = \tilde{\Omega}\left(\frac{(1-\gamma')}{(1-\gamma)}\right)$ then $\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \epsilon\mathbf{1} \leq \boldsymbol{v}^{(T)} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}$.

Proof First, we will induct on t to show that for $t \geq 0$,

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \left(\left(\frac{(\gamma - \gamma')}{(1-\gamma')} \right)^{t} \max_{s \in \mathcal{S}} \left(\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v}^{(0)} \right)(s) + \sum_{j=1}^{t} \left(\frac{(\gamma - \gamma')}{(1-\gamma')} \right)^{(t-j)} \eta \right) \cdot \mathbf{1} \leq \boldsymbol{v}^{(t)} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}. \tag{11}$$

Inductive proof of (11). In the base case, when $t = 0$ the claim reduces to $\mathbf{0} \leq \boldsymbol{v}^{(0)} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}$, which is trivially satisfied because $\boldsymbol{v}^{(0)} = \mathbf{0}$ and $\boldsymbol{r} \in \mathbb{R}^{\mathcal{A}}_{\geq 0}$. For the inductive step, we have

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \left(\left(\frac{(\gamma - \gamma')}{(1-\gamma')} \right)^{t-1} \max_{s \in \mathcal{S}} \left(\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v}^{(0)} \right) + \sum_{j=1}^{t-1} \left(\frac{(\gamma - \gamma')}{(1-\gamma')} \right)^{(t-1-j)} \eta \right) \mathbf{1} \leq \boldsymbol{v}^{(t-1)} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}.$$

Next, observe that

$$\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v}^{(t)} = \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v}^{\star}_{\gamma',\boldsymbol{r}^{(t)}} + \boldsymbol{v}^{\star}_{\gamma',\boldsymbol{r}^{(t)}} - \boldsymbol{v}^{(t)}. \tag{12}$$

By the assumption on $\boldsymbol{v}^{(t)}$,

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma',\boldsymbol{r}^{(t)}} - \boldsymbol{v}^{(t)} \leq \eta\mathbf{1}. \tag{13}$$

By the Bellman optimality conditions, for each $s \in \mathcal{S}$,

$$\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}(s) = \max_{a \in \mathcal{A}_s} \boldsymbol{r}(s,a) + \gamma \boldsymbol{p}(s,a)^{\top} \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} = \max_{a \in \mathcal{A}_s} \boldsymbol{r}(s,a) - (\gamma' - \gamma)\boldsymbol{p}(s,a)^{\top} \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} + \gamma' \boldsymbol{p}(s,a)^{\top} \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}.$$

Consequently, $\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} = \boldsymbol{v}^{\star}_{\gamma',\boldsymbol{r}-(\gamma'-\gamma)\boldsymbol{P}\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}}$. By substituting into (12) and applying the definition of $\boldsymbol{r}^{(t)}$, Lemma 45 implies that

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v}^{(t)} \leq \frac{(\gamma - \gamma')}{(1-\gamma')} \cdot \max_{(s,a) \in \mathcal{A}} (\boldsymbol{P}(\boldsymbol{v}^{(t-1)} - \boldsymbol{v}^{\star}_{\boldsymbol{r},\gamma}))(s,a) \cdot \mathbf{1} + \eta\mathbf{1}$$

$$\leq \frac{(\gamma-\gamma')}{(1-\gamma')} \cdot \max_{s\in\mathcal{S}}(\boldsymbol{v}^{(t-1)}(s) - \boldsymbol{v}^{\star}_{\boldsymbol{r},\gamma}(s)) \cdot \mathbf{1} + \eta\mathbf{1},$$

where in the last step we used that $\|\boldsymbol{P}\|_\infty = 1$ and $\boldsymbol{P}$ is a positive matrix. Consequently,

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \left(\left(\frac{(\gamma-\gamma')}{(1-\gamma')}\right)^t \max_{s\in\mathcal{S}}(\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \boldsymbol{v}^{(0)})(s) + \sum_{j=1}^{t} \left(\frac{(\gamma-\gamma')}{(1-\gamma')}\right)^{t-j} \eta \right) \cdot \mathbf{1} \leq \boldsymbol{v}^{(t)} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}.$$

This completes the inductive argument.

Now, we bound the geometric series as follows

$$\sum_{j=1}^{t} \left(\frac{(\gamma-\gamma')}{(1-\gamma')}\right)^{t-j} \eta = \sum_{j=1}^{t} \left(1 - \frac{(1-\gamma)}{(1-\gamma')}\right)^{t-j} \eta \leq \frac{(1-\gamma')}{(1-\gamma)}\eta.$$

Finally, note that when $T = \tilde{\Omega}\left(\frac{(1-\gamma')}{(1-\gamma)}\right)$ and $\eta \leq \frac{(1-\gamma)}{2(1-\gamma')}\epsilon$,

$$\boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \epsilon \leq \boldsymbol{v}^{(T)} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}.$$

■

Finally, we can leverage Lemma 46 to complete the proof of Theorem 35.

Theorem 35 (PRM - DMDP outer-solver) *Let $\mathcal{M}_{\gamma,\boldsymbol{r}}$ be a DMDP, $\gamma' < \gamma$, and $\epsilon \in (0, 1/(1-\gamma)]$. Let $\epsilon' = \epsilon/4 \cdot (1-\gamma)/(1-\gamma')$. Suppose that in each iteration $t \in [n_{\text{outer}}]$ of Algorithm 4, $\boldsymbol{v}_{t-1/2}$ is a solution to the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}_{t-1}, \epsilon')$-sub-problem. Suppose that $\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1}$ is a solution to the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}_{n_{\text{outer}}}, \epsilon')$-policy sub-problem. Then, $\boldsymbol{v}_{n_{\text{outer}}+1}, \pi_{n_{\text{outer}}+1}$ solves the DMDP problem.*

Proof Note that the outer process in Algorithm 4 (and Algorithm 5) ensures the following. Suppose each $\boldsymbol{v}_{t-1/2}$ is a solution to the $(\mathcal{M}, \boldsymbol{r}, \gamma', \boldsymbol{v}_{t-1}, \epsilon/4 \cdot (1-\gamma)/(1-\gamma'))$-sub-problem. Then due to the post-process, each $\boldsymbol{v}_t$ meets the conditions on $\boldsymbol{v}^{(t)}$ ins Lemma 46 for $\eta = \epsilon/2 \cdot (1-\gamma)/(1-\gamma')$.

Consequently, by Lemma 46, we have that for sufficiently large $n_{\text{outer}} = \tilde{O}((1-\gamma')/(1-\gamma))$,

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \frac{\epsilon}{2}\mathbf{1} \leq \boldsymbol{v}_{n_{\text{outer}}} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}},$$
$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}} - \frac{\epsilon}{2}\mathbf{1} \leq \boldsymbol{v}_{n_{\text{outer}}+1} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}},$$

and by the definition of the policy-sub-problem, we can further conclude

$$\mathbf{0} \leq \boldsymbol{v}^{\star}_{\gamma',\boldsymbol{r}^{n_{\text{outer}}+1}} - \frac{\epsilon}{2}\mathbf{1} \leq \boldsymbol{v}_{n_{\text{outer}}+1} \leq \boldsymbol{v}^{\pi_{n_{\text{outer}}+1}}_{\gamma',\boldsymbol{r}^{n_{\text{outer}}+1}} \leq \boldsymbol{v}^{\star}_{\gamma,\boldsymbol{r}}, \tag{14}$$

where

$$\boldsymbol{r}^{n_{\text{outer}}+1} := \boldsymbol{r} - (\gamma'-\gamma)\boldsymbol{P}\boldsymbol{v}_{n_{\text{outer}}}.$$

Now, note that for all $s \in \mathcal{S}$,

$$\boldsymbol{v}^{\pi_{n_{\text{outer}}+1}}_{\gamma,\boldsymbol{r}}(s) = \boldsymbol{r}(s, \pi_{n_{\text{outer}}+1}(s)) + \gamma \boldsymbol{p}(s, \pi_{n_{\text{outer}}+1}(s))^\top \boldsymbol{v}^{\pi_{n_{\text{outer}}+1}}_{\gamma,\boldsymbol{r}}$$

$$= \boldsymbol{r}(s, \pi_{n_{\text{outer}}+1}(s)) - (\gamma' - \gamma)\boldsymbol{p}(s, \pi_{n_{\text{outer}}+1}(s))^\top \boldsymbol{v}_{\gamma,\boldsymbol{r}}^{\pi_{n_{\text{outer}}}} + \gamma' \boldsymbol{p}(s, \pi_{n_{\text{outer}}+1}(s))^\top \boldsymbol{v}_{\gamma,\boldsymbol{r}}^{\pi_{n_{\text{outer}}+1}}.$$

Above, in the first line we used the Bellman formulation for values of policies (Definition 29).

Thus, $\boldsymbol{v}_{\gamma,\boldsymbol{r}}^{\pi_{n_{\text{outer}}+1}} = \boldsymbol{v}_{\gamma',\boldsymbol{r}-(\gamma'-\gamma)\boldsymbol{P}\boldsymbol{v}_{\gamma,\boldsymbol{r}}^{\pi_{n_{\text{outer}}+1}}}^{\pi_{n_{\text{outer}}+1}}$. Since $(\boldsymbol{I} - \gamma' \boldsymbol{P}^{\pi_{n_{\text{outer}}+1}})^{-1}$ and $\boldsymbol{P}^{\pi_{n_{\text{outer}}+1}}$ are positive matrices and $(\gamma - \gamma')/(1 - \gamma') \le 1$ we have that

$$\begin{aligned}
\boldsymbol{v}_{\gamma',\boldsymbol{r}-(\gamma'-\gamma)\boldsymbol{P}\boldsymbol{v}_{\gamma,\boldsymbol{r}}^{\pi_{n_{\text{outer}}+1}}}^{\pi_{n_{\text{outer}}+1}} - \boldsymbol{v}_{\gamma',\boldsymbol{r}^{n_{\text{outer}}+1}}^{\pi_{n_{\text{outer}}+1}} &= (\boldsymbol{I} - \gamma' \boldsymbol{P}^{\pi_{n_{\text{outer}}+1}})^{-1}((\gamma - \gamma')\boldsymbol{P}^{\pi_{n_{\text{outer}}}}(\boldsymbol{v}_{\gamma,\boldsymbol{r}}^{\pi_{n_{\text{outer}}}} - \boldsymbol{v}_{n_{\text{outer}}})) \\
&\le (\boldsymbol{I} - \gamma' \boldsymbol{P}^{\pi_{n_{\text{outer}}+1}})^{-1}((\gamma - \gamma')\boldsymbol{P}^{\pi_{n_{\text{outer}}}}(\boldsymbol{v}_{\gamma,\boldsymbol{r}}^\star - \boldsymbol{v}^{(n_{\text{outer}})})) \\
&\le \max_{(s,a)\in\mathcal{A}} (\boldsymbol{v}_{\gamma,\boldsymbol{r}}^\star(s,a) - \boldsymbol{v}_{n_{\text{outer}}}(s,a)) \\
&\le (\gamma - \gamma')/(1 - \gamma')\epsilon/2 \\
&\le \frac{\epsilon}{2},
\end{aligned}$$

where the second to last step used that $\|\boldsymbol{I} - \gamma' \boldsymbol{P}^\pi\|_\infty \le 1/(1 - \gamma')$ for any policy π. Consequently, combining with (14), we conclude that $\boldsymbol{v}_{\gamma,\boldsymbol{r}}^\star - \epsilon \le \boldsymbol{v}_{\gamma,\boldsymbol{r}}^{\pi_{n_{\text{outer}}}} \le \boldsymbol{v}_{\gamma,\boldsymbol{r}}^\star$, which completes the proof. ■

5. Application: Matrix games and minimax problems

In this section, we consider minimax problems, including ℓ_2-ℓ_1 and ℓ_2-ℓ_2 matrix-games and finite-sum minimax problems. In Section 5.1 we discuss general preliminaries. In Section 5.2, we discuss ℓ_2-ℓ_1 matrix games in Section 5.2.1 and ℓ_2-ℓ_2 matrix games in Section 5.2.2. Section 5.3 discusses applications of our ℓ_2-ℓ_1 matrix games improvements for two computational geometry problems: maximum inscribed ball and minimum enclosing ball.

5.1. Preliminaries

We first outline preliminaries of the minimax problems we consider. The notation in this subsection is consistent with that of Carmon et al. (2019).

Problem setup. A *setup* is the triplet $(\mathcal{Z} = \mathcal{X} \times \mathcal{Y}, \|\cdot\|, r)$ where we use $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$, where (1) $\mathcal{X}$ is a compact and convex subset of $\mathbb{R}^n$ and $\mathcal{Y}$ is a compact and convex subset of $\mathbb{R}^m$; (2) $\|\cdot\|$ is a norm on $\mathcal{Z}$, and (3) r is a 1-strongly convex function with respect to $\mathcal{Z}$ and $\|\cdot\|$. We can r a *distance generating function* and denote the associated Bregman divergence as

$$V_{\boldsymbol{z}}(\boldsymbol{z}') := r(\boldsymbol{z}') - r(\boldsymbol{z}) - \langle \nabla r(\boldsymbol{z}), \boldsymbol{z}' - \boldsymbol{z} \rangle \ge \frac{1}{2} \left\| \boldsymbol{z}' - \boldsymbol{z} \right\|^2.$$

We also denote $\Theta := \max_{\boldsymbol{z}'} r(\boldsymbol{z}') - \min_{\boldsymbol{z}'} r(\boldsymbol{z})$ and assume it is finite. We use $\|\cdot\|_*$ to denote the dual norm of $\|\cdot\|$. For $\boldsymbol{z} \in \mathcal{Z}$, we often write $\boldsymbol{z}^\mathcal{X} \in \mathcal{X}$, $\boldsymbol{z}^\mathcal{Y} \in \mathcal{Y}$ to denote the first n and last m coordinates of $\boldsymbol{z}$, respectively. We use $d = m + n$ throughout this section.

We assume that $\mathcal{Z}$ is sufficiently simple such that given any $\boldsymbol{z}' \in \mathbb{R}^d$, one can compute the projection of $\boldsymbol{z}'$ onto $\mathcal{Z}$ with respect to $\|\cdot\|$, denoted $\text{proj}_\mathcal{Z}(\boldsymbol{z}')$, in $\tilde{O}(d)$-time. (This is true, for example, for the Euclidean unit ball, or the probability simplex, which are the relevant cases for ℓ_2-ℓ_2 matrix-games and ℓ_2-ℓ_1 matrix games.) Finally, we assume that $c\|\cdot\|_\infty \le \|\cdot\| \le C\|\cdot\|_\infty$ over $\mathcal{Z}$, for some $c, C = \text{poly}(d)$. (This is true, for example, for the ℓ_1 and ℓ_2 norms, which are the relevant cases for ℓ_2-ℓ_2 matrix-games and ℓ_2-ℓ_1 matrix games.)

Minimax problems. We consider minimax (saddle-point) problems of the form

$$\min_{\boldsymbol{x}\in\mathcal{X}} \max_{\boldsymbol{y}\in\mathcal{Y}} f(x,y),$$

for some setup $(\mathcal{Z} = (\mathcal{X},\mathcal{Y}), \|\cdot\|, r)$. We use $g(\boldsymbol{z}) := (\nabla_x f(\boldsymbol{z}), -\nabla_y f(\boldsymbol{z})) \in \mathbb{R}^d$ to denote the *gradient mapping* of f. We use L to denote the Lipschitz constant of g, $D := \max_{\boldsymbol{z},\boldsymbol{z}'\in\mathcal{Z}} \|\boldsymbol{z}-\boldsymbol{z}'\|$, and $G := \max_{\boldsymbol{z}\in\mathcal{Z}} \|g(z)\|$. We assume that D, L, G are finite and hide polylogarithmic dependencies in these parameters inside of $\tilde{O}(\cdot)$ notation.

Next, we define the minimax problem we consider in this Section.

Definition 47 (Minimax problem) *In the minimax problem, we are given $f : \mathcal{Z} = (\mathcal{X}\times\mathcal{Y}) \to \mathbb{R}$, $\epsilon > 0$, and $\delta \in (0,1)$. We must compute $\boldsymbol{x}, \boldsymbol{y}$ such that*

$$\max_{\boldsymbol{y}'\in\mathcal{Y}} f(\boldsymbol{x},\boldsymbol{y}') - \min_{\boldsymbol{x}'\in\mathcal{X}} f(\boldsymbol{x}',\boldsymbol{y}) \le \epsilon.$$

We will later discuss matrix-games and finite-sum minimax problems as special cases of Definition 47 and discuss the relevant batch and sample query models therein. However, in this section we discuss the outer-solver and sub-problem structure for general minimax problems following the conceptual proximal point framework of Carmon et al. (2019); Nemirovski (2004).

Conceptual proximal point. Carmon et al. (2019); Nemirovski (2004) showed how to solve minimax problems of the form of Definition 47 by iteratively solving a series of α-regularized sub-problems. This method is inspired by Nemirovski's "conceptual prox point method" (Nemirovski, 2004). Correspondingly, we define the minimax sub-problem as follows.

Definition 48 (Minimax sub-problem) *Let f be as in Definition 47. Let $\boldsymbol{z}_0 \in \mathcal{Z}, \alpha > 0, \epsilon > 0$. Let $\boldsymbol{z}^\alpha$ be the solution to the problem $\min_{\boldsymbol{x}'\in\mathcal{X}} \max_{\boldsymbol{y}'\in\mathcal{Y}} f(\boldsymbol{z}) + \alpha V_{\boldsymbol{z}}(\boldsymbol{x}',\boldsymbol{y}')$. In the $(\boldsymbol{z}_0,\alpha,\epsilon)$-sub-problem we define $f^{\text{sub}}(\boldsymbol{z}) := \boldsymbol{z}_\alpha$ to be the unique point in $\mathcal{Z}$ such that*

$$\langle g(\boldsymbol{z}_\alpha) + \alpha\nabla V_{\boldsymbol{z}}(\boldsymbol{z}_\alpha), \boldsymbol{z}_\alpha - \boldsymbol{u}\rangle, \quad \textit{for all } \boldsymbol{u}\in\mathcal{Z}.$$

We must output a $\boldsymbol{z}' \in \mathbb{R}^d$ such that $\left\|\boldsymbol{z}' - f^{\text{sub}}(\boldsymbol{z})\right\|_\infty \le \epsilon$.

Carmon et al. (2019); Nemirovski (2004) showed how to solve the minimax problem (Definition 47) by solving a sequence of sub-problems of the form of Definition 48 using Conceptual Proximal Point (CPP) as the outer-solver. Here, the outer process is a projection step (to ensure feasibility) followed by an extragradient step.

Definition 49 (Minimax post-process) *Let f be as in Definition 48, and $\alpha > 0$ For any $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{R}^d \times \mathbb{R}^d$, we define $\boldsymbol{z}'' := \text{proj}_{\mathcal{Z}}(z')$*

$$\zeta(\boldsymbol{z},\boldsymbol{z}') := \operatorname*{argmin}_{\tilde{\boldsymbol{z}}\in\mathcal{Z}} \left(\langle g(\boldsymbol{z}''),\tilde{\boldsymbol{z}}\rangle + \alpha V_{\boldsymbol{z}}(\tilde{\boldsymbol{z}})\right).$$

We now specify the outer-solver framework using CPP as follows. We slightly restate a variant of Proposition 4 of Carmon et al. (2019).

Theorem 50 (CPP, Proposition 4 of Carmon et al. (2019), adapted - Minimax outer-solver) *Let f be as in Definition 47 and $\alpha > 0$. Consider Meta-Algorithm 1 with the following instantiation of parameters. For fixed $\alpha > 0$,*

- *Initialize $\boldsymbol{u}_0$ with $\operatorname{argmin}_{\boldsymbol{z} \in \mathcal{Z}} r(\boldsymbol{z})$.*
- *Define ζ as in Definition 49.*
- *Set $\boldsymbol{w}(t) := 1/n_{\text{outer}}$ for each $t \in [n_{\text{outer}}]$.*

Then, there is an $\epsilon' = \text{poly}(G, L, D, \Theta, \epsilon, d)$ such that the following holds. Suppose that in each iteration $t \in [n_{\text{outer}}]$ of Algorithm 1, $\boldsymbol{u}_{t-1/2}$ is a solution to the $(\boldsymbol{u}_{t-1}, \alpha, \epsilon')$-sub-problem; then $\boldsymbol{u}_{n_{\text{outer}}}$ is a solution to the minimax problem.

Proof To prove this, we appeal to Proposition 4 of Carmon et al. (2019). By Carmon et al. (2019)'s Proposition 4, it is sufficient to show that $\boldsymbol{u}_{t-1/2}$ satisfies

$$\langle g(\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2})), \text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}) - \boldsymbol{u} \rangle - \alpha V_{\boldsymbol{u}_{t-1}}(\boldsymbol{u}) \leq \epsilon \quad \text{for all } \boldsymbol{u} \in \mathcal{Z}. \tag{15}$$

Consequently, to prove the theorem, it suffices to show that whenever $\boldsymbol{u}_{t-1/2}$ is a solution to the $(\boldsymbol{u}_{t-1}, \alpha, \epsilon')$-sub-problem, (15) holds. To this end, let $\boldsymbol{z}_\alpha$ be the unique point in $\mathcal{Z}$ such that

$$\langle g(\boldsymbol{z}_\alpha) + \alpha \nabla V_{\boldsymbol{u}_{t-1}}(\boldsymbol{z}_\alpha), \boldsymbol{z}_\alpha - \boldsymbol{u} \rangle \leq 0, \quad \text{for all } \boldsymbol{u} \in \mathcal{Z}.$$

By the three-point-equality for Bregman divergences, we have that

$$\langle g(\boldsymbol{z}_\alpha), \boldsymbol{z}_\alpha - \boldsymbol{u} \rangle - \alpha V_{\boldsymbol{u}_{t-1}}(\boldsymbol{u}) \leq -\alpha V_{\boldsymbol{z}_\alpha}(\boldsymbol{u}) - \alpha V_{\boldsymbol{u}_{t-1}}(\boldsymbol{z}_\alpha), \quad \text{for all } \boldsymbol{u} \in \mathcal{Z}.$$

Now, since $\|\boldsymbol{u}_{t-1/2} - \boldsymbol{z}_\alpha\|_\infty \leq \epsilon$ and $\boldsymbol{z}_\alpha \in \mathcal{Z}$, we have that

$$\|\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}), \boldsymbol{z}_\alpha\| \leq \|\boldsymbol{u}_{t-1/2}, -\boldsymbol{z}_\alpha\|$$

Consequently, by equivalence of norms,

$$c\|\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}), \boldsymbol{z}_\alpha\|_\infty \leq \|\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}), \boldsymbol{z}_\alpha\| \leq \|\boldsymbol{u}_{t-1/2}, -\boldsymbol{z}_\alpha\| \leq C\|\boldsymbol{u}_{t-1/2}, -\boldsymbol{z}_\alpha\|_\infty.$$

Thus,

$$\|\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}), \boldsymbol{z}_\alpha\|_\infty \leq \|\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}), \boldsymbol{z}_\alpha\| \leq \|\boldsymbol{u}_{t-1/2}, -\boldsymbol{z}_\alpha\| \leq C/c \cdot \|\boldsymbol{u}_{t-1/2}, -\boldsymbol{z}_\alpha\|_\infty.$$

Thus, for any $\boldsymbol{u} \in \mathcal{Z}$ we can write

$$\begin{aligned}
&\langle g(\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2})), \text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}) - \boldsymbol{u} \rangle \\
&= \langle g(\boldsymbol{z}_\alpha), \text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}) - \boldsymbol{u} \rangle + \langle g(\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2})) - g(\boldsymbol{z}_\alpha), \text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2} - \boldsymbol{u} \rangle \\
&= \langle g(\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2})), \boldsymbol{z}_\alpha - \boldsymbol{u} \rangle + \langle g(\boldsymbol{z}_\alpha), \text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}) - \boldsymbol{z}_\alpha \rangle \\
&\quad - \langle g(\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2})) - g(\boldsymbol{z}_\alpha), \text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}) - \boldsymbol{u} \rangle \\
&\leq \langle g(\boldsymbol{z}_\alpha), \boldsymbol{z}_\alpha - \boldsymbol{u} \rangle + \|g(\boldsymbol{z}_\alpha)\|_1 \, C/c \cdot \epsilon' + \|g(\boldsymbol{z}_\alpha) - g(\text{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}))\|_* D
\end{aligned}$$

where in the last line we used Holder's inequality. Now, using the Lipschiztness of g and equivalence of norms,

$$\|g(\boldsymbol{z}_\alpha) - g(\mathrm{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}))\|_* \leq L\|\boldsymbol{z}_\alpha - \boldsymbol{u}_{t-1/2}\| \leq LC\|\boldsymbol{z}_\alpha - \boldsymbol{u}_{t-1/2}\|_\infty \leq LC\epsilon'.$$

And again, using Consequently, taking ϵ' to be a sufficiently small polynomial in L, D, ϵ, d is enough to ensure that (15) holds, i.e., that

$$\langle g(\mathrm{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2})), \mathrm{proj}_{\mathcal{Z}}(\boldsymbol{u}_{t-1/2}) - \boldsymbol{u}\rangle - \alpha V_{\boldsymbol{u}_{t-1}}(\boldsymbol{u}) \leq GC/c \cot \epsilon' + DLC\epsilon' \leq \epsilon.$$

■

In our applications, we often leverage the following fact about Bregman divergences. (Recall that c is defined in Section 5.1).

Fact 51 *Let $r : \mathcal{Z} \to \mathbb{R}$ be a 1-strongly convex function with respect to $\mathcal{Z}$ and $\|\cdot\|$. Then,*

$$V_{\boldsymbol{z}}(\boldsymbol{z}') \geq \frac{1}{2}\left\|\boldsymbol{z}' - \boldsymbol{z}\right\|^2 \geq \frac{c^2}{2}\left\|\boldsymbol{z}' - \boldsymbol{z}\right\|_\infty^2.$$

The specific sub-problem solvers will vary between ℓ_2-ℓ_2 matrix-games, ℓ_2-ℓ_1 matrix-games, and finite-sum minimax problems. However, observe that Theorem 50 already establishes that CPP is robust to bounded ℓ_∞ error in the sub-problem solutions, in the sense of Definition 2. Consequently, it is amenable to using our sample-reuse framework developed in Section 2. We discuss this in the following sections.

5.2. Matrix games

We consider a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and use $\boldsymbol{a}_i$, $\boldsymbol{a}^j$ to denote the i-th row and j-th column of $\boldsymbol{A}$, respectively. We use $A_{i,j}$ to denote the (i,j)-th entry of A. We use $\|\boldsymbol{A}\|_{2\to\infty} := \max_{i\in[m]}\|\boldsymbol{a}_i\|_2$ and $\|\boldsymbol{A}\|_F$ to denote the Frobenius norm.

To discuss the application of our pseudo-independence results for improved oracle complexity trade-offs on matrix games, we first restrict to minimax problems on functions f corresponding to composite matrix games.

Definition 52 (MG problem) *In the (composite) matrix-game problem, we are given a setup $(\mathcal{Z} = \mathcal{X}\times\mathcal{Y}, \|\cdot\|, r)$; a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$; convex, differentiable functions $\phi : \mathcal{X} \to \mathbb{R}, \psi : \mathcal{Y} \to \mathbb{R}$; $\epsilon > 0$; and $\delta \in (0,1)$. We must solve the minimax problem (as in Definition 47) for the function*

$$f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \phi(\boldsymbol{x}) - \psi(\boldsymbol{y}).$$

Next, we define the relevant full batch and sample query oracles for the matrix games problems we consider. As in Carmon et al. (2019) we assume that ϕ and ψ are explicit and that $\boldsymbol{A}$ can be accessed via oracle queries. Concretely, we define one batch oracle and two types of sample oracles for accessing $\boldsymbol{A}$.

Definition 53 (Matrix-vector oracle - MG batch oracle) *When queried with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{Z}$, a matrix-vector oracle for $\boldsymbol{A}$ returns $(\boldsymbol{A}x, \boldsymbol{A}^\top \boldsymbol{y})$.*

Definition 54 (Row/column oracle - MG sample oracle, Type I) *When queried with $(i,)$ for $i \in [m]$, the oracle returns $\boldsymbol{a}_i$, the i-th row of $\boldsymbol{A}$. When queried with $(,j)$ for $j \in [m]$, the oracle returns $\boldsymbol{a}^j$, the j-th column of $\boldsymbol{A}$.*

Definition 55 (Entry oracle - MG sample oracle Type II) *When queried with $(i,j) \in [m] \times [n]$, the entry oracle returns $A_{i,j}$.*

5.2.1. COMPOSITE ℓ_2-ℓ_1 MATRIX GAMES

In this section, we discuss the composite ℓ_2-ℓ_1 matrix games setting. Throughout this section, we use the setup $(\mathcal{Z} = (\mathcal{X}, \mathcal{Y}), \|\cdot\|, r)$ where

- $\mathcal{X} = \mathbb{B}^n := \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_2 \leq 1\}$ is the Euclidean ball of radius 1 centered at the origin; and $\mathcal{Y} = \Delta^m$ is the m-dimensional simplex.
- $\|\cdot\| : \mathcal{Z} \to \mathbb{R}$ is given by $\|\boldsymbol{z}\| = \sqrt{\|\boldsymbol{z}^{\mathcal{X}}\|_2^2 + \|\boldsymbol{y}^{\mathcal{Y}}\|_1^2}$.
- $r : \mathcal{Z} \to \mathbb{R}$ is given by $r(\boldsymbol{z}) = \frac{1}{2}\left\|\boldsymbol{z}^{\mathcal{X}}\right\|_2^2 + \sum_{i\in[m]} \boldsymbol{z}^{\mathcal{Y}}(i)\log(\boldsymbol{z}^{\mathcal{Y}}(i))$.

In this case, the relevant constants defined in Section 5.1 are trivially given by

- $L \leq \|\boldsymbol{A}\|_{2\to\infty}$, $G \leq \max_{i,j}|A_{i,j}|$
- $D = 1$
- $c = 1$, $C = d^2$

The function r is known to be 1-strongly convex with respect to $\mathcal{Z}$, $\|\cdot\|$ (see, e.g., Section 4.2 of Carmon et al. (2019)). To begin, we define the distributions which we will sample from in order to make sample queries.

Definition 56 *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. For each $i \in [m]$, define the distribution $\mathcal{D}_{\text{entry}}(i)$ as follows:*

$$\mathbb{P}_{b\sim\mathcal{D}_{\text{entry}}(i)}\{b = j\} = \frac{A_{i,j}^2}{\|\boldsymbol{a}_i\|_2^2}.$$

The following theorem states the guarantees of the variance-reduced mirror descent (Algorithm 4 of Carmon et al. (2019), VRMD1) solves a minimax-subproblem in the ℓ_2-ℓ_1 matrix-games setting using $\tilde{O}(1)$ full batch queries, a number of non-oblivious, i.e., adaptive sample queries of Type I (see Definition 54), and a number of *oblivious* sample queries of Type II (see Definition 55).

Theorem 57 (VRMD1, Theorem 2 of Carmon et al. (2019), adapted - ℓ_2-ℓ_1 MG sub-solver) *Let f be as defined in Definition 52 and $\boldsymbol{z} \in \mathcal{Z}$. There is a randomized algorithm $\mathcal{A}_{\xi,\chi}^{\text{VRMD1}-\text{HP}|\alpha,\epsilon,\delta}$ such that the following holds.*

- *The algorithm takes in the random seeds ξ, χ distributed as follows, for some sufficiently large $T = \tilde{O}(\|\boldsymbol{A}\|_{2\to\infty}/\alpha^2)$. The first random seed $\xi \sim \{\{(i_{1t}, j_{1t})..., (i_{mt}, j_{mt})\}_{t=1}^T\} \subset [m] \times [n]$ where for each $q \in [m], t \in [T]$, $i_{q_t} = q$ and $j_{q_t} \sim \mathcal{D}_{\text{entry}}(q)$. The second random seed χ is sampled* adaptively *based on $\boldsymbol{z}$ and is used to make some additional* adaptive *sample queries of Type I (row/column oracle queries).*

- *If* $\boldsymbol{z}' = \mathcal{A}_{\xi,\chi}^{\mathrm{VRMD1-HP}|\alpha,\epsilon,\delta}(\boldsymbol{u})$*, then wp.* $1-\delta$ *over the draw of the random seeds* ξ *and* χ*,* $\boldsymbol{z}'$ *solves the* $(\boldsymbol{z}, \alpha, \epsilon)$*-minimax-subproblem.*
- $\mathcal{A}_{\xi,\chi}^{\mathrm{VRMD1-HP}|\alpha,\epsilon,\delta}$ *makes only* $\tilde{O}(1)$ *batch queries, makes sample queries of Type II only the indices encoded in the seed* ξ*, and makes sample queries of Type I* only *on the indices encoded in the seed* χ*.*

Proof The proof follows directly from Theorem 2 of Carmon et al. (2019) and Fact 51. ■

By instantiating CPP (Theorem 50) with VRMD1 (Theorem 57 as the sub-problem solver in the ℓ_2-ℓ_1 setup, we see that we can obtain the following query complexity trade-of of $\tilde{O}(\alpha/\epsilon)$ full batch queries; along with $\tilde{O}(\|A\|_{2\to\infty}^2 \alpha^{-1}\epsilon^{-1})$ non-oblivious sample queries of Type I; and $\tilde{O}(m\|A\|_{2\to\infty}^2 \alpha^{-2})$ oblivious sample queries of Type II. This, corresponds to instantiating Meta-Algorithm 1 with $S = \texttt{Standard}$ and:

- $\boldsymbol{u}_0 = \mathrm{argmin}_{z\in\mathcal{Z}} r(z)$
- $n_{\mathrm{outer}} = \tilde{O}(\alpha/\epsilon)$
- ζ as defined in Definition 49
- $\mathcal{A}_{\xi,\chi}^{\mathrm{VRMD1-HP}}$ as the sub-solver
- $\mathcal{D}_\xi$ and $\mathcal{D}_\chi$ as in Theorem 57
- $f^{\mathrm{sub}}(\boldsymbol{z})$ as defined in Definition 48

Moreover, observe that Theorem 50 ensures that CPP is ℓ_∞ robust with respect to f^{sub} (in the sense of Definition 2), and VRMD1-HP solves sub-problems to high-precision in the ℓ_∞ norm (in the sense of Definition 1). Thus, using our sample-reuse framework, Theorem 6 implies that we can *reuse* the oblivious sample queries of Type II across all n_{outer} iterations of CPP. We obtain the following improved trade-off.

Theorem 58 (ℓ_2-ℓ_1 MG trade-off improvement) *For* $\alpha > 0$*, there is an algorithm that wp.* $1-\delta$ *solves the MG problem in the* ℓ_2-ℓ_1 *setup using* $\tilde{O}(\alpha/\epsilon)$ *full batch queries;* $\tilde{O}(\|\boldsymbol{A}\|_{2\to\infty}^2 \alpha^{-1}\epsilon^{-1})$ *sample queries of Type I; and* $\tilde{O}(m\|\boldsymbol{A}\|_{2\to\infty}^2 \alpha^{-2})$ *sample queries of Type II.*

5.2.2. COMPOSITE ℓ_2-ℓ_2 MATRIX GAMES

In this section, we discuss the composite ℓ_2-ℓ_2 matrix games setting. Throughout this section, we use the setup $(\mathcal{Z} = (\mathcal{X}, \mathcal{Y}), \|\cdot\|, r)$ where

- $\mathcal{X} = \mathbb{B}^n := \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_2 \le 1\}$ and $\mathcal{Y} = \mathbb{B}^m := \{\boldsymbol{y} \in \mathbb{R}^m : \|\boldsymbol{y}\|_2 \le 1\}$ are the Euclidean balls of radius 1 centered at the origin.
- $\|\cdot\| : \mathcal{Z} \to \mathbb{R}$ is given by $\|\boldsymbol{z}\| = \|\boldsymbol{z}\|_2$.
- $r : \mathcal{Z} \to \mathbb{R}$ is given by $r(\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{z}\|_2^2$.

In this case, the relevant constants defined in Section 5.1 are trivially given by

- $L, G \leq \|\boldsymbol{A}\|_2$
- $D = 1$
- $c = 1, C = d$

The function r is known to be 1-strongly convex with respect to $\mathcal{Z}$, $\|\cdot\|$ (see, e.g., Section 4.3 of Carmon et al. (2019)). To begin, we define the distributions which we will sample from in order to make sample queries.

Definition 59 *Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. For each $i \in [m]$, define the distributions $\mathcal{D}_{\text{row}}$ and $\mathcal{D}_{\text{col}}$ as follows:*

$$\mathbb{P}_{a \sim \mathcal{D}_{\text{row}}}\{a = i\} = \frac{\|\boldsymbol{a}_i\|_2^2}{\|\boldsymbol{A}\|_F^2}, \quad \text{and} \quad \mathbb{P}_{a \sim \mathcal{D}_{\text{col}}}\{a = j\} = \frac{\left\|\boldsymbol{a}^j\right\|_2^2}{\|\boldsymbol{A}\|_F^2}.$$

The following theorem states the guarantees of the variance-reduced mirror descent. VRMD-2 solves a subproblem usin $\tilde{O}(1)$ full batch queries and *oblivious* sample queries of Type I (see Definition 54).

Theorem 60 (VRMD2, Lemma 5 of Carmon et al. (2019), adapted - ℓ_2-ℓ_1 MG sub-solver) *Let f be as defined in Definition 52 and $\boldsymbol{z} \in \mathcal{Z}$. There is a randomized algorithm $\mathcal{A}_{\xi,\chi}^{\text{VRMD2-HP}|\alpha,\epsilon,\delta}$ such that the following holds.*

- *The algorithm takes in the random seeds $\xi = (\xi_1, \xi_2)$ distributed as follows, for some sufficiently large $T = \tilde{O}(\|\boldsymbol{A}\|_F^2 / \alpha^2)$. $\xi_1 \sim \{i_1, ..., i_T\} \subset [m]$ and $\xi_2 \sim \{j_1, ..., j_T\} \subset [n]$, where for each $t \in [T]$, $i_t \sim \mathcal{D}_{\text{row}}$ and $j_t \sim \mathcal{D}_{\text{col}}$. The second random seed $\chi \sim \mathsf{Unif}[0]$ is a zero-bit random seed.*
- *If $\boldsymbol{z}' \mathcal{A}_{\xi,\chi}^{\text{VRMD2-HP}|\alpha,\epsilon,\delta}(\boldsymbol{u})$, then wp. $1 - \delta$ over the draw of the random seeds ξ and χ, $\boldsymbol{z}'$ solves the $(\boldsymbol{z}, \alpha, \epsilon)$-minimax-subproblem.*
- *$\mathcal{A}_{\xi,\chi}^{\text{VRMD2-HP}|\alpha,\epsilon,\delta}$ makes only $\tilde{O}(1)$ batch queries, and makes row/column queries (sample queries of Type I) only the indices encoded in the seed ξ. The algorithm makes* no *entry oracle queries (sample queries of Type II).*

Proof The proof follows directly from Lemma 5 of Carmon et al. (2019) and Fact 51. ■

By instantiating CPP (Theorem 50) with VRMD2 (Theorem 60) as the sub-problem solver in the ℓ_2-ℓ_2 setting, we see that we can obtain the following query complexity trade-off of $\tilde{O}(\alpha/\epsilon)$ full batch queries along with $\tilde{O}(m \|A\|_F^2 \alpha^{-1} \epsilon^{-1})$ oblivious sample queries of Type II. This, corresponds to instantiating Meta-Algorithm 1 with $S = \texttt{Standard}$ and:

- $\boldsymbol{u}_0 = \text{argmin}_{z \in \mathcal{Z}} r(z)$
- $n_{\text{outer}} = \tilde{O}(\alpha/\epsilon)$
- ζ as defined in Definition 49
- $\mathcal{A}_{\xi,\chi}^{\text{VRMD2-HP}}$ as the sub-solver

- $\mathcal{D}_\xi$ and $\mathcal{D}_\chi$ as in Theorem 60
- $f^{\text{sub}}(\boldsymbol{z}) := \boldsymbol{z}_\alpha$ as defined in Definition 48

Moreover, observe that Theorem 50 ensures that CPP is ℓ_∞ robust with respect to f^{sub} (in the sense of Definition 2), and VRMD2-HP solves sub-problems to high-precision in the ℓ_∞ norm (in the sense of Definition 1). Thus, using our sample-reuse framework, Theorem 6 implies that we can *reuse* the oblivious sample queries of Type I across all n_{outer} iterations of CPP. We obtain the following improved trade-off.

Theorem 61 (ℓ_2-ℓ_2 MG trade-off improvement) *For $\alpha > 0$, there is an algorithm that wp. $1-\delta$ solves the MG problem in the ℓ_2-ℓ_2 setup using $\tilde{O}(\alpha/\epsilon)$ full batch queries and $\tilde{O}(\|\boldsymbol{A}\|_F^2\,\alpha^{-2})$ sample queries of Type II.*

Optimal Frobenius-norm-dependent query complexities for ℓ_2-ℓ_2 matrix games. In the case of ℓ_2-ℓ_2 games, note that the row/column oracle queries (sample queries of Type I) are *strictly* less powerful than the batch queries (matrix-vector oracle queries) in the sense that one can always use a matrix-vector oracle to implement a row/column oracle. Consequently, setting $\alpha = \|\boldsymbol{A}\|_F^{2/3}\,\epsilon^{1/3}$ in Theorem 61, we obtain an overall matrix-vector oracle query complexity of $\tilde{O}(\|\boldsymbol{A}\|_F^{-2/3}\,\alpha^{-2/3})$.

Note that lower bounds of Liu et al. (2023) indicate that $\tilde{\Omega}(\|\boldsymbol{A}\|_F^{-2/3}\,\alpha^{-2/3})$-matrix-vector oracle queries is information-theoretically necessary. To our knowledge, Theorem 61 is the first to get this information-theoretically near-optimal rate for general ℓ_2-ℓ_2 matrix-games.

5.3. Applications of ℓ_2-ℓ_1 matrix games

In this section, we discuss the implications of our results for two problems in computational geometry. Allen-Zhu et al. (2014); Carmon et al. (2019) showed how to reduce the minimum enclosing ball problem to ℓ_2-ℓ_1 matrix games. Much of the notation and presentation in this section is inspired by Allen-Zhu et al. (2014) and Carmon et al. (2019).

Maximum inscribed ball. In the maximum inscribed ball problem, we are given a polyhedron specified by a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ so that $P = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \geq \boldsymbol{0}\}$. We make the following assumptions, as in Carmon et al. (2019); Allen-Zhu et al. (2014):

- We assume that the polytope P is bounded and hence $m \geq n$.
- We assume that $\|\boldsymbol{a}_i\|_2 = 1$ for all $i \in [m]$ so that $\|\boldsymbol{A}_{2\to\infty}\| = 1$.
- The origin is inside the polytope. This is without loss of generality, as we may always shift the polytope to satisfy this requirement.

In the maximum inscribed ball problem, we must (approximately) compute $\boldsymbol{x}^\star \in P$ such that

$$\boldsymbol{x}^\star \in \operatorname*{argmax}_{\boldsymbol{x}\in P} \min_{i\in[n]} \frac{\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle + b_i}{\|\boldsymbol{a}_i\|_2}. \tag{16}$$

We define

$$r^\star := \max_{\boldsymbol{x}\in P} \min_{i\in[n]} \frac{\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle + b_i}{\|\boldsymbol{a}_i\|_2}, \text{ and}$$

$$R := \min\{r > 0 : P \subset \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_2 \leq r\}\}.$$

We use $\rho := R/r^\star$ to denote the aspect ratio of the problem. Allen-Zhu et al. (2014) showed that solving (16) is equivalent to solving the following minimax problem:

$$\max_{\boldsymbol{x}\in\mathbb{R}^n} \min_{\boldsymbol{y}\in\Delta^m} \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{b}.$$

and formulated the maximum inscribed ball approximation problem as follows.

Definition 62 (Maximum inscribed ball (Max-IB)) *In the* maximum inscribed ball problem*, we must compute $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ such that wp. $1-\delta$,*

$$\min_{\boldsymbol{y}\in\Delta^m} \boldsymbol{y}^\top \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{y}^\top \boldsymbol{b} \geq (1-\epsilon) \max_{\boldsymbol{x}\in\mathbb{R}^n} \min_{\boldsymbol{y}\in\Delta^m} \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{b}.$$

Carmon et al. (2019) further showed that the Max-IB problem can be solved using a two stage approach. In the first stage, we solve ℓ_2-ℓ_1 matrix games of the following form for a sequence of μ's:

$$\max_{\boldsymbol{x}\in\mathbb{B}^n} \min_{\boldsymbol{y}\in\Delta^m} \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{b} + \mu \sum_{i\in[m]} \boldsymbol{y}(i)\log(\boldsymbol{y}(i)) - \frac{\mu}{2}\|\boldsymbol{x}\|_2^2 \tag{17}$$

to obtain a constant-multiplicative approximation $\hat{r}$ to $r^\star$ (Lemma 10 of Carmon et al. (2019)). In the second stage, we solve an ℓ_2-ℓ_1 matrix games of the following form to $O(\epsilon\hat{r})$-accuracy (Theorem 3 of Carmon et al. (2019)):

$$\max_{\boldsymbol{x}\in\mathbb{B}^n} \min_{\boldsymbol{y}\in\Delta^m} \boldsymbol{y}^\top \tilde{\boldsymbol{A}}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{b} \tag{18}$$

where $\tilde{\boldsymbol{A}} = 2R \cdot \boldsymbol{A}$. As our MG Problem definition in the ℓ_2-ℓ_1 setup (Definition 52) captures both (17) and (18), our methods can be used to obtain improved full batch versus sample query complexity trade-offs for solving the Max-IB problem.

Minimum enclosing ball. In the *minimum enclosing ball problem*, we are given a data matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ such that $\boldsymbol{a}_1 = \boldsymbol{0}$, and $\max_{i\in[m]} \|\boldsymbol{a}_i\| = 1$ so that $\|\boldsymbol{A}\|_{2\to\infty} = 1$. We must (approximately) find $R^\star$ such that there exists a point $\boldsymbol{x}$ with $\|\boldsymbol{x} - \boldsymbol{a}_i\|_2 \leq R^\star$ for all $i \in [m]$. That is,

$$R^\star := \min_{\boldsymbol{x}\in\mathbb{R}^n} \max_{\boldsymbol{y}\in\Delta^m} \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{b} + \frac{1}{2}\|\boldsymbol{x}\|_2^2. \tag{19}$$

Definition 63 (Minimum enclosing ball (Min-EB)) *In the minimum enclosing ball problem, we must solve the minimax problem (Definition 47) for $f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{b} + \frac{1}{2}\|\boldsymbol{x}\|_2^2$.*

Carmon et al. (2019) showed that the Min-EB problem can be solved to accuracy $\epsilon/8$ by solving the following ℓ_2-ℓ_1 matrix game to accuracy $\epsilon/16$ (Lemma 11 of Carmon et al. (2019)):

$$\min_{\boldsymbol{x}\in\mathbb{B}^n} \max_{\boldsymbol{y}\in\Delta^m} \boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{b} - \frac{\epsilon}{32\log(m)} \sum_{i\in[m]} \boldsymbol{y}(i)\log(\boldsymbol{y}(i)) + \frac{1}{2}\|\boldsymbol{x}\|_2^2. \tag{20}$$

As our MG Problem definition in the ℓ_2-ℓ_1 setup (Definition 52) captures both (20) our methods can be used to obtain improved full batch versus sample query complexity trade-offs for solving the Min-EB problem.

6. Application: Finite-sum minimization with non-uniform smoothness

In this section, we discuss applications of pseudo-independence for improved full batch versus sample query trade-offs for finite sum minimization (FSM) where the component functions are of non-uniform smoothness. This is a generalization of Section 3.

Definition 64 (Generalized FSM (GFSM) problem) *In the GFSM problem, we are given $c > 1$ and $\boldsymbol{x}_0 \in \mathbb{R}^d$ and must output $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ such that $F(\hat{\boldsymbol{x}}) - \min_{\boldsymbol{x}} F(\boldsymbol{x}) \leq 1/c \cdot (F(\boldsymbol{x}_0) - \min_{\boldsymbol{z}} F(\boldsymbol{x}))$ where $F : \mathbb{R}^d \to \mathbb{R}$ with $F : \mathbb{R}^d \to \mathbb{R}$ witj $F(x) := \frac{1}{n}\sum_{i\in[n]} f_i(x)$, F is μ strongly-convex, and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is of known smoothnes L_i.*

State-of-the-art query complexities for non-uniform smoothness finite-sum minimization can be achieved by using a primal-dual extra-gradient method (Jin et al., 2022). By using this primal-dual extra-gradient method as our sub-problem solver for solving regularized problems and using APP as the outer-solver (Frostig et al., 2015) as in the uniform-smoothness case (Section 3), we can obtain trade-offs between batch and sample queries that depend on the distribution of the smoothness parameters L_i (rather than bounds that depend only on the worst-case smoothness L as in Section 3).

The gradient (batch) oracle and component (sample) oracle are the same as for FSM (Definitions 16 and 17. The FSM sub-problems and f^{sub} are defined exactly as in the uniform smoothness case (Definition 18.) The only change relative to Section 3 is the choice of sub-solver. For the GFSM problem, we use the primal-dual finite-sum minimization algorithm of Jin et al. (2022).

Theorem 65 (PDFSM, Theorem 2 of Jin et al. (2022), restated - GFSM sub-problem solver) *Let F be as in Definition 64. There is a randomized algorithm $\mathcal{A}_{\xi,\chi}^{\text{PDFSM}|\lambda,c,\delta}(\boldsymbol{x})$ such that the following holds.*

- *The algorithm takes in the random seeds ξ, χ distributed as follows. The first random seed $\xi \sim \{i_1, ..., i_T\}$ where each $\mathbb{P}[i_t = j] = \sqrt{L_j}/(\sum_{k\in[n]} \sqrt{L_k})$ for some*

$$T = \tilde{O}\left(\sum_{i\in[n]} \sqrt{\frac{L_i}{n\mu}}\right).$$

  The second random seed $\chi \sim \mathsf{Unif}[0]$ is a 0-bit random seed.

- *If $\boldsymbol{x}' = \mathcal{A}_{\xi,\chi}^{\text{PDFSM}|\lambda,c,\delta}(\boldsymbol{u})$, then wp. $1-\delta$ over the draw of the random seeds ξ and χ, $\boldsymbol{x}'$ is a solution to the $(\boldsymbol{u}, \lambda, c)$-sub-problem.*

- *$\mathcal{A}_{\xi,\chi}^{\text{PDFSM}|\lambda,c,\delta}$ makes only $\tilde{O}(1)$ batch queries and makes sample queries* only *on the indices contained in xi.*

As in the nonuniform case, by combining Theorem 65 with Theorem 20, we obtain a standard trade-off of $\tilde{O}(\sqrt{\lambda/\mu})$ batch (gradient oracle) queries and $\tilde{O}\left(\sqrt{\lambda/\mu} \cdot \sum_{i\in[n]} \sqrt{\frac{L_i}{n\mu}}\right)$ sample (component oracle) queries for any $\lambda \geq \mu$.

However, as in Section 3, using our sample-reuse framework, we observe that we can *reuse* randomness across all $n_{\text{outer}} = \sqrt{\lambda/\mu}$ sub-problem solves. More concretely, using identical convexity argument as in the proof of Lemma 23, we obtain the following analog of Theorem 65.

Lemma 66 (GFSM sub-problem solver high-precision) *Let F be as in Definition 64. There is a randomized algorithm $\mathcal{A}_{\xi,\chi}^{\text{PDFSM-HP}|\lambda,c,\delta}(\boldsymbol{x})$ such that the following holds.*

- *The algorithm takes in the random seeds ξ, χ distributed as follows. The first random seed $\xi \sim \{i_1, ..., i_T\}$ where each $\mathbb{P}[i_t = j] = \sqrt{L_j}/(\sum_{k\in[n]} \sqrt{L_k})$ for some*

$$T = \tilde{O}\left(\sum_{i\in[n]} \sqrt{\frac{L_i}{n\mu}}\right).$$

  The second random seed $\chi \sim \mathsf{Unif}[0]$ is a 0-bit random seed.

- *If $\boldsymbol{x}' = \mathcal{A}_{\xi,\chi}^{\text{SVRG-HP}|\lambda,c,\delta}(\boldsymbol{u})$, then wp. $1-\delta$ over the draw of the random seeds ξ and χ,*

$$\left\|\boldsymbol{x}' - \bar{f}_{\lambda'}^{\text{sub}}(\boldsymbol{y_u})\right\|_\infty^2 \le \frac{1}{c} \cdot \left(F(\boldsymbol{y_u}) - \min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^d} F(\tilde{\boldsymbol{x}}) + \frac{\lambda}{2}\left\|\tilde{\boldsymbol{x}} - \boldsymbol{y_u}\right\|_2^2\right).$$

- *$\mathcal{A}_{\xi,\chi}^{\text{PDFSM-HP}|\lambda,c,\delta}$ makes only $\tilde{O}(1)$ batch queries and makes sample queries* only *on the indices contained in ξ.*

By combining Lemma 66 with Lemma 22 and applying Theorem 6, we obtain a trade-off of $\tilde{O}(\sqrt{\lambda/\mu})$ batch (gradient oracle) queries and $\tilde{O}\left(\sqrt{\lambda/\mu} \cdot \sum_{i\in[n]} \sqrt{\frac{L_i}{n\mu}}\right)$ sample (component oracle) queries for any $\lambda \ge \mu$. The pseudo-code is analogous to Algorithm 3 except that the inner-loop sub-problem solver is replaced with PDFSM-HP in place of SVRG-HP.

Theorem 67 (Non-uniform smoothness FSM trade-off improvement) *For $\lambda \ge \mu$, there is an algorithm that solves FSM with non-uniform smoothness (Definition 64) using $\tilde{O}(\sqrt{\lambda/\mu})$ full batch queries and only $\tilde{O}(\sum_{i\in[n]} \sqrt{L_i/(n\lambda)})$ sample queries.*

7. Application: Top eigenvector computation

Here, we discuss the setting of top eigenvector (TopEV) computation. This is an interesting specialized setting in which our improvement for finite-sum optimization (Definition 15) can be applied even if the components f_i of the finite sum might not be convex.

Throughout this section, we use $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ to denote a matrix, and we use $\boldsymbol{a}_1, ..., \boldsymbol{a}_n \in \mathbb{R}^d$ to denote its rows. We use $\|\boldsymbol{A}\|_F := \sqrt{\sum_{i,j} \boldsymbol{A}_{i,j}^2}$ and $\|\boldsymbol{A}\|_2$ to denote the spectral norm of $\boldsymbol{A}$. We use $\text{sr}(\boldsymbol{A}) := \sum_i \frac{\lambda_i}{\lambda_1} = \|\boldsymbol{A}\|_F^2 / \|\boldsymbol{A}\|_2^2$, where $\lambda_1 \ge \lambda_2 \ge \cdots \lambda_d$ are the eigenvalues of $\boldsymbol{\Sigma}$. Note that we always have $\text{sr}(\boldsymbol{A}) \le \text{rank}(\boldsymbol{A})$. We define the relative eigen-gap $\text{gap}(\boldsymbol{A}) := \frac{\lambda_1 - \lambda_2}{\lambda_1}$.

Definition 68 (TopEV problem) *In the TopEV problem, we are given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and $\epsilon > 0$ and must compute a unit vector $\boldsymbol{x} \in \mathbb{R}^d$ such that $\boldsymbol{x}^\top \boldsymbol{\Sigma} \boldsymbol{x} \ge (1-\epsilon)\lambda_1$ where $\boldsymbol{\Sigma} := \boldsymbol{A}^\top \boldsymbol{A} \in \mathbb{R}^{d\times d}$ and λ_1 is the largest eigenvalue of $\boldsymbol{\Sigma}$.*

In typical settings (Garber et al., 2016), the goal is to solve this problem with probability inverse polynomial in d (i.e., with high probability in d.) We define the batch and sample queries for solving the problem as follows.

Definition 69 (Matrix-vector oracle - TopEV batch oracle) *When queried with $\boldsymbol{x} \in \mathbb{R}^d$, the* matrix-vector *oracle returns $\boldsymbol{A}\boldsymbol{x}$.*

Definition 70 (Row oracle - TopEV sample oracle) *When queried, with $i \in [n]$, a* row oracle *for $\boldsymbol{A}$ returns $\boldsymbol{a}_i \in \mathbb{R}^d$.*

Reducing top eigenvector computation to solving linear systems in $\lambda' \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A}$ Garber et al. (2016) showed how to reduce top eigenvector computation to performing an approximate version of the classical inverse power method in the shifted matrix $\lambda \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A}$, where λ is an appropriately chosen parameter. This is called the *approximate shift-and-invert* power method and can be implemented given access to an oracle that approximately solves linear systems in $\lambda \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A}$.

Furthermore, Garber et al. (2016) also showed that selecting $(1 + \mathrm{gap}(\boldsymbol{A})/150)\lambda_1 \leq \lambda \leq (1 + \mathrm{gap}(\boldsymbol{A})/100)\lambda_1$ is sufficient to ensure that $\tilde{O}(1)$ iterations of approximate shift-and-invert power method is sufficient to solve the top eigenvector problem. Section 6 of Garber et al. (2016) shows that given an algorithm for approximately solving linear systems in $\lambda' \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A}$ for $\lambda' > \lambda_1 + \mathrm{gap}(\boldsymbol{A})/120$, there is a method for computing a valid shift parameter λ (such that $(1 + \mathrm{gap}(\boldsymbol{A})/150)\lambda_1 \leq \lambda \leq (1 + \mathrm{gap}(\boldsymbol{A})/100)\lambda_1$) with runtime and query complexity overhead that is only lower order relative to the approximate shift-and-invert power method steps.

Consequently, in the remainder of this section, we simply focus on the problem of solving linear systems of the form

$$(\lambda' \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A})\boldsymbol{x} = \boldsymbol{b},$$

where $\lambda' > \lambda_1 + \Theta(\mathrm{gap}(\boldsymbol{A}))$ and $\boldsymbol{b}$ is known, since this is the fundamental subroutine used in the algorithms of Garber et al. (2016). Concretely, we summarize their reductions as follows.

Theorem 71 (Theorems 5, 8, 15, and 30 of Garber et al. (2016)) *Given an algorithm* `Solve`$(\boldsymbol{x}_0 \boldsymbol{A}, \lambda', \boldsymbol{b})$ *that computes $\boldsymbol{x}$ such that wp. $1 - \mathrm{poly}(1/d, \mathrm{gap}(\boldsymbol{A}))$,*

$$\left\| \boldsymbol{x} - (\lambda' \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A})^{-1} \boldsymbol{b} \right\|_2^2 \leq \mathrm{poly}(1/d, \mathrm{gap}(\boldsymbol{A})) \left\| \boldsymbol{x}_0 - (\lambda' \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A})^{-1} \boldsymbol{b} \right\|_2^2,$$

for any $\lambda > \lambda_1 + \mathrm{gap}(\boldsymbol{A})/120$, there is an algorithm that solves the TopEV problem with only $\tilde{O}(1)$ calls to `Solve`.

Thus, in the remainder of this section, we consider the problem of solving linear systems of the form

$$(\lambda' \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A})\boldsymbol{x} = \boldsymbol{b}, \tag{21}$$

for some $\boldsymbol{b} \in \mathbb{R}^n$ and $\lambda' > \lambda_1 + \mathrm{gap}(\boldsymbol{A})/120$. This can equivalently be viewed as the following minimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{2} \boldsymbol{x}^\top (\lambda' \boldsymbol{I} - \boldsymbol{A}^\top \boldsymbol{A})\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x} = \min_{\boldsymbol{x}} \frac{1}{n} \sum_{i \in [n]} \frac{1}{2} \cdot \boldsymbol{x}^\top (w_i \boldsymbol{I} - n \boldsymbol{a}_i \boldsymbol{a}_i^\top)\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}, \tag{22}$$

whenever $\sum_i w_i/n = \lambda'$. This is reminiscent of the FSM problem (Definition 15) with $f_i(\boldsymbol{x}) = \frac{1}{2} \cdot \boldsymbol{x}^\top (w_i \boldsymbol{I} - n \boldsymbol{a}_i \boldsymbol{a}_i^\top)\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}$. However, we have the caveat that the matrices $(w_i \boldsymbol{I} - \boldsymbol{a}_i \boldsymbol{a}_i^\top)$ need

not be positive semi-definite; and consequently, the f_i's are not necessarily convex *even though* F *is* $\lambda' - \lambda_1 = \Theta(\text{gap}(\boldsymbol{A}))$*-strongly convex* (and $\|A\|_F^2$-smooth).

Interestingly, the outer-solver procedure APP (Theorem 20) still applies in this setting; however, the guarantees of SVRG (Theorem 21) unfortunately do not apply *directly* when the f_i are potentially non-convex. Nonetheless, Garber et al. (2016) leveraged problem structure to show that SVRG can be applied for (22). Consequently, we define the TopEV sub-problem similarly as in Definition 21 for $F(\boldsymbol{x}) := f_i(\boldsymbol{x})$, where

$$F(\boldsymbol{x}) := f_i(\boldsymbol{x}) \text{ where } f_i(x) := \frac{1}{2} \cdot \boldsymbol{x}^\top \left(w_i \boldsymbol{I} - n \boldsymbol{a}^{(i)} \boldsymbol{a}^{(i)\top} \right) \boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}; w_i := n \cdot \frac{\lambda' \|\boldsymbol{a}_i\|_2^2}{\|\boldsymbol{A}\|_F^2}. \quad (23)$$

Note that from this perspective, the batch oracle call for TopEV (Definition 16) exactly corresponds to a gradient oracle call for F, and a sample oracle for call for F exactly corresponds to a component oracle call for F (Definition 17.)

Definition 72 (TopEV sub-problem) *Let F be as in (23) and $\rho \geq \lambda' - \lambda_1$. In the $(\boldsymbol{u}, \rho, c)$-sub-problem for TopEV, we must solve the $(\boldsymbol{u}, \rho, c)$-FSM-sub-problem (Definition 18.)*

Note the similarity between the TopEV sub-problem and the FSM sub-problem from Definition 18. The only difference is that the f_i need not be *convex*. Nonetheless, Garber et al. (2016) provide the following guarantee, which is analogous to Theorem 21 from the FSM setting in Section 3.

Theorem 73 (SVRG, Theorem 2.2 of Frostig et al. (2015), restated - TopEV sub-problem solver) *Let F be as defined in (23) and $\rho \geq (\lambda' - \lambda_1)$. There is a randomized algorithm $\mathcal{A}_{\xi,\chi}^{\text{SVRG}-\text{TopEV}|\rho,c,\delta}$ such that the following holds.*

- *The algorithm takes in the random seeds ξ, χ distributed as follows. The first random seed $\xi \sim \{i_1, ..., i_T\}$ where each $\mathbb{P}(i_j = k) = \|\boldsymbol{a}_k\|_2^2 / \|\boldsymbol{A}\|_F^2$ for some $T = \tilde{O}(L/\lambda)$. The second random seed $\chi \sim \text{Unif}[0]$ is a 0-bit random seed.*
- *If $\boldsymbol{x}' = \mathcal{A}_{\xi,\chi}^{\text{SVRG}-\text{TopEV}|\rho,c,\delta}(\boldsymbol{u})$, then wp. $1 - \delta$ over the draw of the random seeds ξ and χ, $\boldsymbol{x}'$ is a solution to the $(\boldsymbol{u}, \rho, c)$-sub-problem.*
- *$\mathcal{A}_{\xi,\chi}^{\text{SVRG}-\text{TopEV}|\rho,c,\delta}$ makes only $\tilde{O}(1)$ batch queries and makes sample queries* only *on the indices encoded in ξ.*

By Theorem 20, we can instantiate APP using SVRG (Theorem 73) with the to solve problems of the form (21) as required by Theorem 71. The pseudocode is the same as Algorithm 2, replacing SVRG-HP with SVRG-TopEV. Setting δ to be inversely polynomial in d, this solves the TopEV problem with high probability in d and obtains a trade-off of $\tilde{O}(\sqrt{\rho/(\lambda' - \lambda_1)})$ full batch (matrix-vector oracle) queries and

$$\tilde{O}\left(\frac{\rho^2 + 12\lambda_1 \|\boldsymbol{A}\|_F^2}{(\rho - \lambda_1 + \lambda)^2} \right) \cdot \sqrt{\frac{\rho}{\lambda' - \lambda_1}}$$

sample (row oracle) queries for any $\rho \geq \lambda' - \lambda_1$.

Now, since F in this case remains smooth and strongly convex (even though the f_i's may not be convex), using the *exact* same arguments as in Section 3 (Lemmas 22 and Lemma 23) we can use our sample reuse framework from Section 2 (Theorem 6) to obtain the following improved trade-off. (Again, th pseudocode is the same as Algorithm 3, replacing SVRG-HP with SVRG-TopEV.)

Theorem 74 (TopEV trade-off improvement) *For $\lambda \geq \mu$, there is an algorithm that solves FSM with high probability in d using only $\tilde{O}(\sqrt{\rho/(\lambda' - \lambda_1)})$ full batch queries and only $\tilde{O}\left(\frac{\rho^2 + 12\lambda_1 \|\boldsymbol{A}\|_F^2}{(\rho - \lambda_1 + \lambda)^2}\right)$ sample queries.*

Finally, we remark that to obtain the trade-offs reported in Table 2, we simply make the change of variables $\rho = \alpha\lambda_1$. With this change of variables,

$$\sqrt{\frac{\rho}{\lambda' - \lambda_1}} = \sqrt{\frac{\alpha}{\mathrm{gap}(\boldsymbol{A})}}, \quad \text{and} \quad \frac{\rho^2 + 12\lambda_1 \|\boldsymbol{A}\|_F^2}{(\rho - \lambda_1 + \lambda)^2} = \tilde{O}\left(\frac{\alpha^2\lambda_1^2 + 12\lambda_1 \|\boldsymbol{A}\|_F^2}{\alpha^2\lambda_1^2}\right) = \tilde{O}\left(\frac{\mathrm{sr}(\boldsymbol{A})}{\alpha^2}\right).$$

8. Conclusion

We introduced a sample reuse framework to reuse randomness across multiple subproblem solutions in variance-reduced optimization methods without sacrificing theoretical correctness guarantees. Our results enabled improved query complexity trade-offs for a broad range of optimization problems. One limitation of our current sample reuse framework and analysis of pseudo-independence is that it only allows us to reuse samples across *high-accuracy* subroutines. Although this already enables improvements for several problems, as seen in Table 2, in future work it may be interesting to study whether tighter analysis allows sample reuse even for sub-routines which do not solve to high-accuracy. This could have applications, for instance, to non-convex optimization problems. As mentioned in Section 1.1, another limitation is that our results don't directly yield a worst-case asymptotic-runtime improvement that we are aware of; however it sheds new light on the information needed to solve FSM and could yield faster algorithms depending on caching and memory layout. Building on this approach to obtain efficiency gains, e.g., in distributed computing settings, is an interesting direction for future research.

Acknowledgments

We thank anonymous reviewers for their feedback. Yujia Jin and Ishani Karmarkar were funded in part by NSF CAREER Award CCF-1844855, NSF Grant CCF-1955039, and a PayPal research award. Aaron Sidford was funded in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF1955039, and a PayPal research award. Yujia Jin's contributions to the project occurred while she was a graduate student at Stanford.

References

Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. In *32nd International Conference on Machine Learning (ICML)*, 2015.

Zeyuan Allen-Zhu, Zhenyu Liao, and Yang Yuan. Optimization algorithms for faster computational geometry. In *41st International Colloquium on Automata, Languages and Programming (ICALP)*, 2014.

Hilal Asi and John C Duchi. Near instance-optimality in differential privacy. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Amos Beimel, Haim Kaplan, Yishay Mansour, Kobbi Nissim, Thatchaphol Saranurak, and Uri Stemmer. Dynamic algorithms against an adaptive adversary: generic constructions and lower bounds. In *54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022.

Dimitri P Bertsekas. Approximate policy iteration: A survey and some new methods. In *Journal of Control Theory and Applications*, 2011.

Vladimir Braverman, Robert Krauthgamer, Aditya Krishnan, and Shay Sapir. Lower bounds for pseudo-deterministic counting in a stream. *arXiv preprint arXiv:2303.16287*, 2023.

Yair Carmon, Yujia Jin, Aaron Sidford, and Kevin Tian. Variance reduction for matrix games. *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.

Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022.

Yeshwanth Cherapanamjeri and Jelani Nelson. On adaptive distance estimation. In *Advances in Neural Information Processing Systems*, 2020.

Edith Cohen, Jelani Nelson, Tamás Sarlós, Mihir Singhal, and Uri Stemmer. One attack to rule them all: Tight quadratic bounds for adaptive queries on cardinality sketches. In *arXiv preprint arXiv:2411.06370*, 2024.

Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *6th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2015.

Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Journal of the ACM*, 2020a.

Michael B Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. In *Theory of Computing*, 2020b.

Peter Dixon, Aduri Pavan, Jason Vander Woude, and NV Vinodchandran. Pseudodeterminism: promises and lowerbounds. In *54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022.

Matteo Fischetti, Iacopo Mandatelli, and Domenico Salvagnin. Faster sgd training by minibatch persistency. In *arXiv preprint arXiv:1806.07353*, 2018.

Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *32nd International Conference on Machine Learning (ICML)*, 2015.

Dan Garber, Elad Hazan, Chi Jin, Cameron Musco, Praneeth Netrapalli, Aaron Sidford, et al. Faster eigenvector computation via shift-and-invert preconditioning. In *33rd International Conference on Machine Learning (ICML)*, 2016.

Erann Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. In *Electronic Colloquium on Computational Complexity: ECCC*, 2011.

Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Jelani Nelson. Differentially private all-pairs shortest path distances: Improved algorithms and lower bounds. In *33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2022.

Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic nc. In *44th International Colloquium on Automata, Languages and Programming (ICALP)*, 2017.

Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. Pseudo-deterministic proofs. In *arXiv preprint arXiv:1706.04641*, 2017.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. In *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 2012.

Ofer Grossman, Meghal Gupta, and Mark Sellke. Tight space lower bound for pseudo-deterministic approximate counting. In *64th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2023.

Russell Impagliazzo, Rex Lei, Toniann Pitassi, and Jessica Sorrell. Reproducibility in learning. In *Proceedings of the 54th annual ACM SIGACT symposium on theory of computing*, 2022.

Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general lps. In *53rd Annual ACM Symposium on Theory of Computing (STOC)*, 2021.

Yujia Jin and Aaron Sidford. Towards tight bounds on the sample complexity of average-reward mdps. In *38th International Conference on Machine Learning (ICML)*, 2021.

Yujia Jin, Aaron Sidford, and Kevin Tian. Sharper rates for separable minimax and finite sum optimization via primal-dual extragradient methods. In *35th Annual Conference on Computational Learning Theory (COLT)*, 2022.

Yujia Jin, Ishani Karmarkar, Aaron Sidford, and Jiayi Wang. Truncated variance reduced value iteration. In *arXiv preprint arXiv:2405.12952*, 2024.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26 (NeurIPS)*, 2013.

Ishani Karmarkar, Liam O'Carroll, and Aaron Sidford. Solving zero-sum games with fewer matrix-vector products. In *Foundations of Computer Science*, 2025.

Michael Kearns and Satinder Singh. Finite-sample convergence rates for q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 1998.

Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014.

Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*, 2015.

Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *11th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 1995.

Yuanshi Liu, Hanzhen Zhao, Yang Xu, Pengyun Yue, and Cong Fang. Accelerated gradient algorithms with adaptive subspace search for instance-faster optimization. In *arXiv preprint arXiv:2312.03218*, 2023.

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*, 2004.

Sebastien Roch. Modern discrete probability: An essential toolkit. In *Cambridge Series in Statistical and Probabilistic Mathematics*, 2024.

Yousef Saad. Numerical methods for large eigenvalue problems: revised edition. In *SIAM*, 2011.

Scott Sallinen, Nadathur Satish, Mikhail Smelyanskiy, Samantika S Sury, and Christopher Ré. High performance parallel stochastic gradient descent in shared memory. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016.

Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2018a.

Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018b.

Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. In *Naval Research Logistics (NRL)*, 2023.

Paul Tseng. Solving h-horizon, stationary markov decision problems in time proportional to log (h). In *Operations Research Letters*, 1990.

Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and l1-regression in nearly linear time for dense instances. In *53rd Annual ACM Symposium on Theory of Computing (STOC)*, 2021.

Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P. Liu, Richard Peng, and Aaron Sidford. Faster maxflow via improved dynamic spectral vertex sparsifiers. In *54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022.

Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, 2016.

Blake E Woodworth, Kumar Kshitij Patel, and Nati Srebro. Minibatch vs local sgd for heterogeneous distributed learning. In *Advances in Neural Information Processing Systems*, 2020.

Appendix A. Inducing pseudoindependence numerically stably

The pseudo-independent algorithm constructed in the proof of Theorem 12 is simple; however, it may not be directly implementable in finite precision, as it requires infinite precision to directly implement the addition of uniform random noise. The proof of Theorem 75 below provides an alternative construction that can be implemented in finite precision.

Theorem 75 (Finite precision analog of Theorem 12) *Let $\epsilon, \delta \in (0,1)$, $\eta > 0$, $\eta' := \min\left(\frac{\eta}{2}, \frac{\eta\epsilon}{p}\right)$, and let $\mathcal{A}_{\xi,\chi}$ be a randomized algorithm that is an (η', δ)-approximation of a function $f : \mathbb{R}^d \to \mathbb{R}^p$. There is a numerically stable algorithm $\mathcal{A}'_{\xi,\chi'}$ such that $\mathcal{A}'_{\xi,\chi'}$ is an (ϵ, δ)-pseudo-independent of ξ and an (η, δ)-approximation of f with the* same *runtime and query complexities as $\mathcal{A}_{\xi,\chi}$ up to an additive $O(p)$ in runtime.*

Proof Consider $\mathcal{S} \subset \mathbb{R}^p$ to be a β-covering of $\mathbb{R}^p$ in the ℓ_∞ norm. Define $\mathsf{round} : \mathbb{R}^p \to \mathcal{S}$ to be the operator that maps $\boldsymbol{x} \in \mathbb{R}^p$ to some $\boldsymbol{x}' \in \mathbb{R}^p$ such that $0 \leq \boldsymbol{x}(i) - \boldsymbol{x}'(i) \leq \beta$ for all $i \in [d]$, Define $\mathsf{smooth}_{\boldsymbol{\nu},t}$ to be the operator which uses a random seed $\boldsymbol{\nu} \sim \mathcal{D}_{\boldsymbol{\nu}}$ to map $\boldsymbol{x} \in \mathcal{S}$ to a uniformly random $\boldsymbol{x}' \in \{y \in \mathcal{S} : -t \leq (\boldsymbol{x}(i) - \boldsymbol{x}'(i)) \leq t\}$. Here, t is a parameter that will be specified later in the proof.

Let $\mathcal{A}'_{\xi,\chi'}$ be the randomized algorithm which takes input $\boldsymbol{x} \in \mathbb{R}^d$, random seed $\xi \sim \mathcal{D}_\xi$, and χ where $\chi = (\chi', \boldsymbol{\nu}) \sim \mathcal{D}_\chi^{\boldsymbol{x}}$ is the concatenation of an independently drawn seed $\chi' \sim \mathcal{D}_{\chi'}^{\boldsymbol{x}}$ and seed $\boldsymbol{\nu} \sim \mathcal{D}_{\boldsymbol{\nu}}$. For any realization s, c, n of $\xi, \chi', \boldsymbol{\nu}$, let $\mathcal{A}'_{\xi=s,\chi'=(c,\boldsymbol{e})}(\boldsymbol{x}) = \mathsf{smooth}_{\boldsymbol{\nu}=\boldsymbol{e},t}(\mathsf{round}(\mathcal{A}_{\xi=s,\chi=c}(\boldsymbol{x})))$.

First, we'll construct a smoothing for $\mathcal{A}'_{\xi,\chi'}$. Let $\bar{\mathcal{A}}_{\chi'}$ be the randomized algorithm which takes input $\boldsymbol{x} \in \mathbb{R}^d$ and a random seed $\chi' \sim \mathcal{D}_{\chi'}^{\boldsymbol{x}}$. For any realization c, n of $\chi', \boldsymbol{\nu}$, we define the mapping $\bar{\mathcal{A}}_{\chi'=(c,\boldsymbol{e})}(\boldsymbol{x}) = \mathsf{smooth}_{\boldsymbol{\nu}=\boldsymbol{e},t}(\mathsf{round}(f(\boldsymbol{x})))$. Now, using that $\mathcal{A}_{\xi,\chi}$ is an (η', δ)-approximation of f, we can conclude that

$$\mathbb{P}_{\xi\sim\mathcal{D}_\xi, \chi\sim\mathcal{D}_\chi^{\boldsymbol{x}}}\left(\|\mathcal{A}_{\xi,\chi}(\boldsymbol{x}) - f(\boldsymbol{x})\|_\infty \leq \eta'\right) \geq 1 - \delta, \tag{24}$$

and hence

$$\mathbb{P}_{\xi\sim\mathcal{D}_\xi, \chi'\sim\mathcal{D}_{\chi'}^{\boldsymbol{x}}}\left(\|\mathcal{A}'_{\xi,\chi'}(\boldsymbol{x}) - f(\boldsymbol{x})\|_\infty \leq (\eta' + t)\right) \geq 1 - \delta. \tag{25}$$

Thus, $\mathcal{A}'_{\xi,\chi'}$ is an $(\eta' + t, \delta)$-approximation of f. In particular, setting $t = \tau = \max\left(\frac{\eta'}{2}, \frac{\eta' p}{2\epsilon}\right)$ and recalliing $\eta' = \min\left(\frac{\eta}{2}, \frac{\eta\epsilon}{p}\right)$, we have

$$\eta' + t \leq \eta' \max\left(\frac{3}{2}, \frac{p}{\epsilon}\right) \leq \eta$$

and hence $\mathcal{A}'_{\xi,\chi'}$ is an (η,δ)-approximation of f. To bound the total variation distance, note that by (24), we have that with probability $1-\delta$, $d_{TV}\left(p_{A_{\xi=s,\chi}(x)}, p_{\bar{A}_\chi(\boldsymbol{x})}\right)$ is the total variation distance between q, q' where q is the distribution of a $\mathsf{smooth}_{\boldsymbol{\nu},t}(0)$ and q' is the distribution of $\mathsf{smooth}_{\boldsymbol{\nu},t}(\boldsymbol{\alpha})$ where $\boldsymbol{\alpha}$ is ℓ_∞ bounded by η'. Using that $t \geq \eta'/2$, we have that

$$\begin{aligned} d_{TV}\left(q,q'\right) &= 1 - \prod_{i\in[p]} \left(\frac{\max\left(0, \left\lfloor \frac{2t-|\alpha_i|}{\beta} \right\rfloor \cdot \beta\right)}{\lceil 2t/\beta \rceil \cdot \beta} \right) \leq 1 - \prod_{i\in[p]} \frac{\left\lfloor \frac{2t-\eta'}{\beta} \right\rfloor}{\lceil 2t/\beta \rceil} \leq 1 - \left(\frac{2t-\eta'-\beta}{2t+\beta}\right)^p \\ &= 1 - \left(1 - \frac{\eta'-2\beta}{2t+\beta}\right)^p \leq \left(\frac{\eta'+2\beta}{2t}\right) p \end{aligned}$$

where the final step used Bernoulli's inequality and that $\beta \geq 0$. Thus, setting $t = \tau = \max\left(\frac{\eta'}{2}, \frac{(\eta'+2\beta)p}{2\epsilon}\right)$, we have that with probability $1-\delta$,

$$\mathbb{P}_{s\sim\mathcal{D}_\chi}\left\{ d_{TV}\left(p_{A'_{\xi=s,\chi'}(x)}, p_{\bar{A}_\chi(\boldsymbol{x})}\right) \leq \epsilon \right\} \geq 1-\delta.$$

For the second guarantee, note that $\mathcal{A}'_{\xi,\chi'}$ has the same runtime and query complexities as $\mathcal{A}_{\xi,\chi}$ up to an additive $O(p)$ increase in the runtime due to the cost of performing the p-dimensional random perturbation induced by $\boldsymbol{\nu}$. ■

Appendix B. Pseudoindependence and repeated compositions

In this section, our goal is to prove the following theorem (see Definition 13 for related notation.)

Lemma 76 *Let $\mathcal{A}'_{\xi,\chi'}$ be randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and two random seeds $\xi \sim \mathcal{D}_\xi, \chi' \sim \mathcal{D}^{\boldsymbol{u}}_{\chi'}$ and is (ϵ,δ)-pseudo-independent of ξ. Then,*

$$d_{TV}\left(p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};s,\mathsf{D}_{\chi'})}, p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})}\right) \leq 2T(\delta+\epsilon)\,.$$

Before proving Lemma 14, we state the following fact about transformations of random variables.

Fact 77 *Let A and B be random variables in $\mathcal{Q}$ and let ζ be a deterministic function $\zeta : \mathcal{Q} \to \mathcal{Q}$. Then, $d_{TV}\left(p_{\zeta(A)}, p_{\zeta(B)}\right) \leq d_{TV}\left(p_A, p_B\right)$.*

Proof For any deterministic function f, let $f^{-1}(\cdot)$ denote the preimage of f. That is, for any $T \subset \mathcal{Q}$, let $f^{-1}(T) := \{\omega' \in \Omega : f(\omega') \in T\}$. Then, by the definition of the total variation distance, we have

$$\begin{aligned} d_{TV}\left(p_{\zeta(A)}, p_{\zeta(B)}\right) &= \sup_{S\subset\mathcal{Q}} \left|\, \mathbb{P}\left\{\zeta(A)\in S\right\} - \mathbb{P}\left\{\zeta(B)\in S\right\} \,\right| \\ &= \sup_{S\subset\mathcal{Q}} \left|\, \mathbb{P}\left\{A\in \zeta^{-1}(S)\right\} - \mathbb{P}\left\{B\in \zeta^{-1}(S)\right\} \,\right| \\ &\leq \sup_{T\subset\mathcal{Q}} \left|\, \mathbb{P}\left\{A\in T\right\} - \mathbb{P}\left\{B\in T\right\} \,\right| = d_{TV}\left(p_A, p_B\right). \end{aligned}$$

■

Now, to prove Lemma 14, we first bound the TV distance between $p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})}$ and $p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})}$ (recall Definition 13.)

Lemma 78 *Let $\mathcal{A}'_{\xi,\chi'}$ be a randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and two random seeds $\xi \sim \mathcal{D}_\xi, \chi' \sim \mathcal{D}^{\boldsymbol{u}}_{\chi'}$. Suppose $\mathcal{A}'_{\xi,\chi'}$ is (ϵ,δ)-pseudo-independent and that $\bar{\mathcal{A}}_{\chi'}$ is an (ϵ,δ)-smoothing of $\mathcal{A}'$ with respect to ξ. Let $s \sim \mathcal{D}_\xi$. Then,*

$$d_{TV}\left(p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})}, p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})}\right) \le T(\delta+\epsilon).$$

Proof Induct on T. If $T = 1$, the statement essentially follows from the definition of pseudo-independence, Fact 7, and a union bound, as we elaborate in the next few sentences. Concretely, we have that with probability $1-\delta$ over the draw of $s \sim \mathcal{D}_\xi$,

$$d_{TV}\left(p_{\mathcal{A}'_{\xi=s,\chi'}(\boldsymbol{u})}, p_{\bar{\mathcal{A}}_{\chi'}(\boldsymbol{u})}\right) \le \epsilon.$$

Since ζ is a deterministic function, using Fact 77, we have

$$d_{TV}\left(p_{\zeta(\boldsymbol{u},\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u}))}, p_{\zeta(\boldsymbol{u},\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u}))}\right) \le d_{TV}\left(p_{\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u})}, p_{\mathcal{A}_{\xi=s,\chi}(\boldsymbol{u})}\right) \le \epsilon.$$

Consequently, by Fact 7 and union bound,

$$d_{TV}\left(p_{\Phi^1_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})}, p_{H^1_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})}\right) \le (\delta+\epsilon).$$

This completes the base case.

For the inductive step, suppose the theorem holds up to $T-1$. That is, suppose that

$$d_{TV}\left(p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})}, p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})}\right) \le (T-1)(\delta+\epsilon).$$

Then, by Fact 7, there exists an event E_1 and random variables C, D, F such that conditioned on E_1, $\Phi^{T-1}_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'}) \overset{\mathcal{D}}{=} C \overset{\mathcal{D}}{=} H^{T-1}_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})$ and $\mathbb{P}\{E_1\} \ge 1-(T-1)(\delta+\epsilon)$. Consequently, for any event E, we have

$$\begin{aligned}\left|p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})}(E) - p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})}(E)\right| &\le \left|p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})|E_1}(E) - p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})|E_1}(E)\right| + \mathbb{P}\{\neg E_1\} \\ &= \left|p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})|E_1}(E) - p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})|E_1}(E)\right| + (T-1)(\delta+\epsilon)\end{aligned}$$

To bound the first term, we observe that by the definition of Φ and H, we have

$$\begin{aligned}&\left|p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_\chi)|E_1}(E) - p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_\chi)|E_1}(E)\right| \\ &= \left|p_{\zeta\left(\Phi^{T-1}_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'}),\mathcal{A}'_{\xi,\chi'}\left(\Phi^{T-1}_{\mathcal{A}'}(\boldsymbol{u};\mathcal{D}_\xi,\mathsf{D}_{\chi'})\right)\right)|E_1}(E) - p_{\zeta\left(H^{T-1}_{\mathcal{A}'}(\boldsymbol{u};\mathsf{D}_{\chi'}),\bar{\mathcal{A}}_{\xi,\chi'}\left(H^{T-1}_{\bar{\mathcal{A}}}(\boldsymbol{u};\mathsf{D}_{\chi'})\right)\right)|E_1}(E)\right| \\ &\le \sup_{\boldsymbol{c}\in\mathbb{R}^d}\left|p_{\zeta\left(\boldsymbol{c},\mathcal{A}'_{\xi,\chi'}(\boldsymbol{c})\right)} - p_{\zeta\left(\boldsymbol{c},\bar{\mathcal{A}}_{\chi'}(\boldsymbol{c})\right)}\right|,\end{aligned}$$

where in the last line, we used the fact that conditional on E_1, $\Phi^{T-1}(x; \mathcal{D}_\xi, \mathsf{D}_{\chi'}) \overset{\mathcal{D}}{=} C \overset{\mathcal{D}}{=} H^{T-1}(x; \mathsf{D}_{\chi'})$. By Fact 77 and the definition of pseudo-independence (using an identical argument to the base case) we have

$$\sup_{\boldsymbol{c}\in\mathbb{R}^d} \left| p_{\zeta\left(\boldsymbol{c}, \mathcal{A}'_{\xi,\chi'}(\boldsymbol{c})\right)} - p_{\zeta(\boldsymbol{c}, \bar{\mathcal{A}}_{\chi'}(\boldsymbol{c}))} \right| \leq (\delta + \epsilon).$$

Thus, by substitution, we conclude that for any event E,

$$\left| p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_\chi)|E_1}(E) - p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_\chi)|E_1}(E) \right| \leq (\delta + \epsilon),$$

and consequently,

$$\left| p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_{\chi'})}(E) - p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_{\chi'})}(E) \right| \leq T(\delta + \epsilon).$$

■

The following lemma obtains the analogous bound as Lemma 78 when the same random seed ξ is reused across iterations.

Lemma 79 *Let $\mathcal{A}'_{\xi,\chi'}$ be a randomized algorithm which takes an input $x \in \mathbb{R}^d$ and two random seeds $\xi \sim \mathcal{D}_\xi, \chi' \sim \mathcal{D}_{\chi'}{}^{\boldsymbol{u}}$. Let $\bar{\mathcal{A}}_{\chi'}$ be a (ϵ, δ)-smoothing of $\mathcal{A}'$ with respect to ξ. Let $s \sim \mathcal{D}_\xi$. Then,*

$$d_{TV}\left(p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'})}, p_{H^T_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_{\chi'})} \right) \leq T(\delta + \epsilon).$$

Proof By Fact 7 and Fact 77, with probability $1 - T\delta$ over the draw of s, there exist random variables C^t and events E_t for $t = \{1, ..., T\}$ so that $\mathbb{P}\{E_t\} \geq 1 - \epsilon$, and conditioned on $E_1, ..., E_t$,

$$\zeta(\boldsymbol{u}, \mathcal{A}'_{\xi=s,\chi'}(\boldsymbol{u})) \overset{\mathcal{D}}{=} C^1 \overset{\mathcal{D}}{=} \zeta(\boldsymbol{u}, \bar{\mathcal{A}}_{\chi'}(\boldsymbol{u})),$$

and

$$\Phi^t_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'}) \overset{\mathcal{D}}{=} \zeta(C^{t-1}, \mathcal{A}_{\xi=s,\chi'}(C^{t-1})) \overset{\mathcal{D}}{=} C^t,$$
$$H^t_{\bar{\mathcal{A}}}(\boldsymbol{u}; \mathsf{D}_{\chi'}) \overset{\mathcal{D}}{=} \zeta(C^{t-1}, \bar{\mathcal{A}}_{\chi'}(C^{t-1})) \overset{\mathcal{D}}{=} C^t.$$

for every $t \in [T]$. So, let $E' := \wedge_{t\in[T]} E_t$, and note that $\mathbb{P}\{E'\} \geq 1 - \epsilon T$. Consequently,

$$d_{TV}\left(p_{\Phi^T(x; s, \mathsf{D}_{\chi'})}, p_{H^T(x; \mathsf{D}_{\chi'})} \right) \leq T\delta + (1 - \mathbb{P}\{E'\}) \leq T(\delta + \epsilon).$$

■

Lemma 80 *Let $\mathcal{A}'_{\xi,\chi'}$ be randomized algorithm which takes an input $\boldsymbol{u} \in \mathbb{R}^d$ and two random seeds $\xi \sim \mathcal{D}_\xi, \chi' \sim \mathcal{D}^{\boldsymbol{u}}_{\chi'}$ and is (ϵ, δ)-pseudo-independent of ξ. Then,*

$$d_{TV}\left(p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; s, \mathsf{D}_{\chi'})}, p_{\Phi^T_{\mathcal{A}'}(\boldsymbol{u}; \mathcal{D}_\xi, \mathsf{D}_{\chi'})} \right) \leq 2T(\delta + \epsilon).$$

Proof The result follows directly by applying triangle inequality, Lemma 79, and Lemma 78. ■