RECAPP: Crafting a More Efficient Catalyst for Convex Optimization

Yair Carmon [1] **Arun Jambulapati** [2] **Yujia Jin** [2] **Aaron Sidford** [2]

Abstract

The accelerated proximal point algorithm (APPA), also known as "Catalyst", is a well-established reduction from convex optimization to approximate proximal point computation (i.e., regularized minimization). This reduction is conceptually elegant and yields strong convergence rate guarantees. However, these rates feature an extraneous logarithmic term arising from the need to compute each proximal point to high accuracy. In this work, we propose a novel Relaxed Error Criterion for Accelerated Proximal Point (RECAPP) that eliminates the need for high accuracy subproblem solutions. We apply RECAPP to two canonical problems: finite-sum and max-structured minimization. For finite-sum problems, we match the best known complexity, previously obtained by carefully-designed problem-specific algorithms. For minimizing $\max_y f(x, y)$ where f is convex in x and strongly-concave in y, we improve on the best known (Catalyst-based) bound by a logarithmic factor.

1 Introduction

A fundamental approach to optimization algorithm design is to break down the problem of minimizing $F : \mathcal{X} \to \mathbb{R}$ into a sequence of easier optimization problems, whose solution converges to $x^\star \in \arg\min_{x\in\mathcal{X}} F(x)$. A canonical "easier" problem is proximal point computation, i.e., computing

$$\mathrm{prox}_{F,\lambda}(x) = \operatorname*{arg\,min}_{x'\in\mathcal{X}} \left\{ F(x') + \frac{\lambda}{2} \|x' - x\|^2 \right\}.$$

Here $\lambda > 0$ is a regularization parameter which balances between how close computing $\mathrm{prox}_{F,\lambda}$ is to minimizing F, and how easy it is to compute—as λ decreases $\mathrm{prox}_{F,\lambda}(x)$ tends to the true minimizer but becomes harder to compute.

[1]Tel Aviv University [2]Stanford University. Correspondence to: Yujia Jin <yujiajin@stanford.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

The classical proximal point method (Rockafellar, 1976; Parikh & Boyd, 2014) simply iterates $x_{t+1} = \mathrm{prox}_{F,\lambda}(x_t)$, and (for convex F) minimizes F with rate $F(x_T) - F(x^\star) = O\big(\frac{\lambda R^2}{T}\big)$ for $R := \|x_0 - x^\star\|$. This rate can be improved, at essentially no additional cost, by carefully combining proximal steps, i.e. $\mathrm{prox}_{F,\lambda}(x_t)$, with gradient steps using $\nabla F(\mathrm{prox}_{F,\lambda}(x_t)) = \lambda(x_t - \mathrm{prox}_{F,\lambda}(x_t))$ (Güler, 1992). This accelerated method converges with rate $O\big(\frac{\lambda R^2}{T^2}\big)$, a quadratic improvement over proximal point.

To turn this conceptual acceleration scheme into a practical algorithm, one must prescribe the accuracy to which each proximal point needs to be computed. Güler (1992); Salzo & Villa (2012) provided such conditions, which were later refined in independently-proposed accelerated approximate proximal point algorithm (APPA) (Frostig et al., 2015) and Catalyst (Lin et al., 2015; 2017). Furthermore, APPA/Catalyst obtained global convergence guarantees for concrete problems by using linearly convergent algorithms to compute the proximal points to their required accuracy. The APPA/Catalyst framework has since been used to accelerate full-batch gradient descent, coordinate methods, finite-sum variance reduction (Frostig et al., 2015; Lin et al., 2017), eigenvalue problems (Garber et al., 2016), min-max problems (Yang et al., 2020), and more.

However, the simplicity and generality of APPA/Catalyst seems to come at a practical and theoretical cost: satisfying existing proximal point accuracy conditions requires solving subproblems to fairly high accuracy. In practice, this means expending computation in subproblem solutions that could otherwise be used for more outer iterations. In theory, APPA/Catalyst complexity bounds feature a logarithmic term that appears unnecessary. For example, in the finite sum problem of minimizing $F(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ where each f_i is convex and L-smooth, APPA/Catalyst combined with SVRG (Johnson & Zhang, 2013) and $\lambda = L/n$ finds an ϵ-optimal point of F in $O\Big((n + \sqrt{nLR^2/\epsilon}) \log \frac{LR^2}{\epsilon}\Big)$ computations of ∇f_i, a nearly optimal rate (Woodworth & Srebro, 2016). A line of work devoted to designing accelerated method tailored to finite sum problems (Shalev-Shwartz & Zhang, 2014; Allen-Zhu, 2016; Lan et al., 2019) attains progressively better practical performance and theoretical guarantees, culminating in an $O\Big(n \log\log n + \sqrt{nLR^2/\epsilon}\Big)$ complexity bound (Song et al., 2020).

Given the power of APPA/Catalyst, it is natural to ask whether the additional logarithmic complexity term is fundamentally tied to the black-box structure that makes it generally-applicable? Indeed, Lin et al. (2017) speculate that the logarithmic term "may be the price to pay for a generic acceleration scheme."

Our work proves otherwise by providing a new Relaxed Error Condition for Accelerated Proximal Point (RECAPP) which standard subproblem solvers can satisfy without incurring an extraneous logarithmic complexity term. For finite-sum problems, our approach combined with SVRG recovers the best existing complexity bound of $O\Big(n \log\log n + \sqrt{nLR^2/\epsilon}\Big)$.[1] Preliminary experiments on logistic regression problem indicate that our method is competitive with Catalyst-SVRG in practice.[2]

As an additional application of our framework, we consider the problem of minimizing $F(x) = \max_{y\in\mathcal{Y}} f(x,y)$ for a function f that is L-smooth, convex in x and μ-strongly-convex in y. The best existing complexity bound for this problem is $O\Big(\frac{LR}{\sqrt{\mu\epsilon}} \log \frac{LR}{\sqrt{\mu\epsilon}}\Big)$ evaluations of ∇f, using an extension of APPA/Catalyst to min-max problems (Yang et al., 2020), and mirror-prox (Korpelevich, 1976; Nemirovski, 2004; Azizian et al., 2020) as the subproblem solver.[3] Our framework (also combined with mirror-prox) removes this logarithmic factor, finding a point with expected suboptimality ϵ in $O\Big(\frac{LR}{\sqrt{\mu\epsilon}}\Big)$ gradient queries (up to lower order terms), which is asymptotically optimal (Ouyang & Xu, 2021). We summarize our complexity bounds in Table 1.

Technical overview. Our development consists of four key parts. First, we define a criterion on the function-value error of the proximal point computation (Definition 3.1) that significantly relaxes the relative error conditions of prior work; see Section 3.3 for a detailed comparison. Second, instead of directly bounding the distance error of the approximate proximal points (as most prior works implicitly do), we follow Asi et al. (2021) and require an *unbiased* estimator of the proximal point whose variance is bounded similarly to the function-value error (Definition 3.2). We prove that any approximation satisfying these guarantees has the same convergence bounds on its (expected) error as the exact accelerated proximal points method. Third, we use the multilevel Monte Carlo technique (Giles, 2015; Blanchet & Glynn, 2015; Asi et al., 2021) to obtain the required unbiased proximal point estimator using (in expectation) a constant number of queries to any method satisfying the function-value error criterion. Finally, we show how to use SVRG and mirror-prox to efficiently meet our error criterion, allowing us to solve finite-sum and minimax optimization problems without the typical extra logarithmic factors incurred by previous proximal point frameworks.

Even though we maintain the same iteration structure as APPA/Catalyst, our novel error criterion induces two nontrivial modifications to the algorithm. First and foremost, our relaxed error bound depends on the previous approximate proximal point x_{t-1} as well as the current query point s_{t-1} (see Algorithm 1). This dependence strongly suggests that the subproblem solver should depend on x_{t-1} somehow. For finite-sum problems we use x_{t-1} as the reference point for variance reduction, while for max-structured problems we initialize mirror-prox with $x = s_{t-1}$ and (approximately) $y = \arg\max_{y\in\mathcal{Y}} f(x_{t-1}, y)$. The second algorithmic consequence, which appeared previously in Asi et al. (2021), stems from the fact that our function-value error and zero-bias/bounded-variance requirements are leveraged for distinct parts of the algorithm (the prox step and gradient step, respectively). This naturally leads to using distinct approximate prox points for each part: one directly obtained from the subproblem solver and one debiased via MLMC.

Paper organization. After providing some notation and preliminaries in Section 2, we present our improved inexact accelerated proximal point framework in Section 3. We then instantiate our framework: in Section 4 we consider finite-sum problems and SVRG (providing preliminary empirical results in Section 4.1) and in Section 5 we consider min-max problems and mirror-prox. We provide additional discussion of related work, including recent independent work by Kovalev & Gasnikov (2022), in Appendix A. The rest of the appendix is composed of the proofs for each corresponding section, followed by Appendix F which provides a discussion of limitations and possible extensions of this work.

2 Preliminaries

General notation. Throughout, $\mathcal{X}$ and $\mathcal{Y}$ refer to closed, convex sets, with diameters denoted by R and R' respectively (when needed). We use F to denote a convex function defined on $\mathcal{X}$. For any parameter $\lambda > 0$ and point $s \in \mathcal{X}$, we let

$$F_s^\lambda(x) := F(x) + \frac{\lambda}{2}\|x - s\|^2 \tag{1}$$

denote the proximal regularization of F around x, and let $\mathrm{prox}_{F,\lambda}(s) = \arg\min_{x\in\mathcal{X}} F_s^\lambda(x)$.

[1] Our framework gives accelerated linear convergence for strongly-convex objectives via a standard restarting argument; see Proposition 3.6.

[2] Code available at: `github.com/yaircarmon/recapp`.

[3] Yang et al. (2020) also establish the same rate for the dual problem of maximizing $\Psi(y) = \min_{x\in\mathcal{X}} f(x,y)$ over y. Since our acceleration framework is primal-only, we are currently unable to remove the logarithmic factor from that rate.

Objective F	Reg.	ApproxProx complexity	WarmStart complexity	Overall complexity	Ref.
general	λ	$\mathcal{T}_{\text{A}}$	$\mathcal{T}_{\text{W}}$	$O\left(\mathcal{T}_{\text{A}}\sqrt{\frac{\lambda R^2}{\epsilon}}+\mathcal{T}_{\text{W}}\right)$	Thm. 3.5
				$O\left(\mathcal{T}_{\text{A}}\sqrt{\frac{\lambda}{\gamma}}\log\frac{LR^2}{\epsilon}+\mathcal{T}_{\text{W}}\right)$	Prop. 3.6
$\frac{1}{n}\sum_{i\in[n]}f_i(x)$	$\frac{L}{n}$	$O(n)$	$O(n\log\log n)$	$O\left(\sqrt{\frac{nLR^2}{\epsilon}}+n\log\log n\right)$ $\widetilde{O}\left(\sqrt{\frac{nL}{\gamma}}\right)+O(n\log\log n)$	Thm. 4.3
$\max_{y\in\mathcal{Y}} f(x,y)$	μ	$O\left(\frac{L}{\mu}\right)+\widetilde{O}\left(\sqrt{\frac{L}{\mu}}\right)$	$\widetilde{O}\left(\frac{L}{\mu}\right)$	$O\left(\frac{LR}{\sqrt{\mu\epsilon}}\right)+\widetilde{O}\left(\frac{L}{\mu}+\sqrt{\frac{LR^2}{\epsilon}}\right)$ $\widetilde{O}\left(\frac{L}{\sqrt{\mu\gamma}}+\frac{L}{\mu}\right)$	Thm. 5.4

Table 1: **Summary of our results.** Throughout, ϵ denotes the solution accuracy, and γ denotes the strong convexity parameter of F. For finite sum problems (the middle row) we assume that each f_i is L-smooth, and measure complexity in terms of ∇f_i evaluations. For max-structured problem (the bottom) we assume that f is L-smooth and μ-strongly-concave in y, and measure complexity in ∇f evaluations. The notation $\widetilde{O}(\cdot)$ hides a logarithmic term. See Section 3 for the definition of ApproxProx and WarmStart.

Distances and norms. We consider Euclidean space throughout the paper and use $\|\cdot\|$ to denote standard Euclidean norm. We denote a *projection* of $x \in \mathbb{R}^d$ onto a closed subspace $\mathcal{X} \subseteq \mathbb{R}^d$ by $\mathsf{Proj}_{\mathcal{X}}(x) = \arg\min_{x'\in\mathcal{X}}\|x-x'\|$. For a convex function $F:\mathcal{X}\to\mathbb{R}$, we denote the *Bregman divergence* induced by F as $V_x^F(x') := F(x') - F(x) - \langle\nabla F(x), x'-x\rangle$, for every $x, x' \in \mathcal{X}$. We denote the Euclidean Bregman divergence by $V_x^{\text{e}}(x') := V_x^{\frac{1}{2}\|\cdot\|^2}(x') = \frac{1}{2}\|x'-x\|^2$.

Smoothness, convexity and concavity. Given a differentiable, convex function $F:\mathcal{X}\to\mathbb{R}$, we say F is *L-smooth* if its gradient $\nabla F:\mathcal{X}\to\mathcal{X}^*$ is L-Lipschitz. We say F is *μ-strongly-convex* if for all $x, x'\in\mathcal{X}$, $F(x')\geq F(x)+\langle\nabla F(x), x'-x\rangle+\frac{\mu}{2}\|x'-x\|^2$. A function Ψ is μ-strongly concave if $-\Psi$ is μ-strongly convex. For $f(x,y)$ that is convex in x and concave in y, the point $(x^\star, y^\star)$ is a saddle-point if $\max_{y\in\mathcal{Y}} f(x^\star,y)\leq f(x^\star,y^\star)\leq\min_{x\in\mathcal{X}} f(x,y^\star)$ for all $x,y\in\mathcal{X}\times\mathcal{Y}$.

3 Framework

In this section, we present our Relaxed Error Criterion Accelerated Proximal Point (RECAPP) framework. We start by defining our central algorithms and relaxed error criteria (Section 3.1). Next, we state our main complexity bounds (Section 3.2) and sketch its proof. Then, we illustrate our new relaxed error criterion by comparing it to the error requirements of prior work (Section 3.3). Finally, as an illustrative warm-up, we show our framework easily recovers the complexity bound of Nesterov's classical accelerated gradient descent (AGD) method (Nesterov, 1983).

3.1 Methods and Key Definitions

Algorithm 1 describes our core accelerated proximal method. The algorithm follows the standard template of the (inexact) accelerated proximal point method, except that unlike most such methods (but similar to the methods of Asi et al. (2021)), our algorithm relies on two distinct approximations of $\text{prox}_{F,\lambda}(s_t)$ with different relaxed error criterion. We now define each approximation in turn.

Our first relaxed error criterion constrains the function value of the approximate proximal point and constitutes our key contribution.

Definition 3.1 (ApproxProx)**.** *Given convex function $F:\mathcal{X}\to\mathbb{R}$, parameter $\lambda>0$, and points $s, x_{\text{init}}, x_{\text{prev}}\in\mathcal{X}$, the point $x = \text{ApproxProx}_{F,\lambda}(s; x_{\text{init}}, x_{\text{prev}})$ is an approximate minimizer of $F_s^\lambda(x) := F(x)+\lambda V_s^{\text{e}}(x)$ such that for $x^\star := \text{prox}_{F,\lambda}(s) = \arg\min_{x\in\mathcal{X}} F_s^\lambda(x)$,*

$$\mathbb{E}F_s^\lambda(x) - F_s^\lambda(x^\star) \leq \frac{\lambda V_{x^\star}^{\text{e}}(x_{\text{init}}) + V_{x^\star}^F(x_{\text{prev}})}{8}. \quad (2)$$

Beyond the prox-center s, our robust error criterion depends

Algorithm 1: RECAPP

Parameters: $\lambda > 0$, step budget T
Initialize $\alpha_0 \leftarrow 1$ and
$x_0 = v_0 \leftarrow \textsc{WarmStart}_{F,\lambda}(R^2)$
▷ To satisfy $\mathbb{E}F(x_0) - F(x^\star) \le \lambda R^2$
for $t = 0$ **to** $T-1$ **do**
  Set $\alpha_{t+1} \in [0,1]$ to satisfy $\frac{1}{\alpha_{t+1}^2} - \frac{1}{\alpha_{t+1}} = \frac{1}{\alpha_t^2}$
  $s_t \leftarrow (1-\alpha_{t+1})\, x_t + \alpha_{t+1} v_t$
  $x_{t+1} \leftarrow \textsc{ApproxProx}_{F,\lambda}(s_t; s_t, x_t)$
  $\widetilde{x}_{t+1} \leftarrow \textsc{UnbiasedProx}_{F,\lambda}(s_t; x_t)$
  $v_{t+1} \leftarrow \mathsf{Proj}_{\mathcal{X}}\big(v_t - \frac{1}{\alpha_{t+1}}(s_t - \widetilde{x}_{t+1})\big)$
Return: x_T

Algorithm 2: UnbiasedProx via MLMC

Input: ApproxProx, points $s, x_{\text{prev}} \in \mathcal{X}$
Parameter: Geometric distribution parameter $p \in [0,1)$ and integer offset $j_0 \ge 0$
Outut: Unbiased estimator of $x^\star = \text{prox}_{F,\lambda}(s)$
$x^{(0)} \leftarrow \textsc{ApproxProx}_{F,\lambda}(s; s, x_{\text{prev}})$
Sample $J_+ \sim \mathsf{Geom}\,(1-p) \in \{0,1,2,\ldots\}$
$J \leftarrow j_0 + J_+$
for $j = 0$ **to** $J-1$ **do**
  $x^{(j+1)} \leftarrow \textsc{ApproxProx}_{F,\lambda}(s; x^{(j)}, x^{(j)})$
$p_J \leftarrow \mathbb{P}[\mathsf{Geom}\,(1-p) = J_+] = (1-p)\cdot p^{J_+}$
Return: $x^{(j_0)} + p_J^{-1}(x^{(J)} - x^{(\max\{J-1, j_0\})})$

on two additional points: x_{init} (which in Algorithm 1 is also set the prox-center s_t) and x_{prev} (which in Algorithm 1 is set to the previous iterate x_t). The criterion requires the suboptimality of the approximate solution to be bounded by weighted combination of two distances: the Euclidean distance between the true proximal point $x^\star$ and x_{init}, and the Bregman divergence (induced by F) between $x^\star$ and x_{prev}. In Section 3.3 we provide a detailed comparison between our criterion and prior work, but note already that—unlike APPA/Catalyst—the relative error we require in (2) is *constant*, i.e., independent of the desired accuracy or number of iterations. This constant level of error is key to enabling our improved complexity bounds.

Our second relaxed error criterion constrains the bias and variance of the approximate proximal point.

Definition 3.2 (UnbiasedProx)**.** *Given convex function $F : \mathcal{X} \to \mathbb{R}$, parameter $\lambda > 0$, and points s and $x_{\text{prev}} \in \mathcal{X}$, the point $x = \textsc{UnbiasedProx}_{F,\lambda}(s; x_{\text{prev}})$ is an approximate minimizer of $F_s^\lambda(x) = F(x) + \lambda V_s^{\text{e}}(x)$ such that $\mathbb{E}\, x = x^\star = \text{prox}_{F,\lambda}(s) = \arg\min_{x\in\mathcal{X}} F_s^\lambda(x)$, and*

$$\mathbb{E}\,\|x - x^\star\|^2 \le \frac{\lambda V_{x^\star}^{\text{e}}(s) + V_{x^\star}^{F}(x_{\text{prev}})}{4\lambda}. \tag{3}$$

Note that the any $x = \textsc{ApproxProx}_{F,\lambda}(s; s, x_{\text{prev}})$ satisfies the distance bound (3) (due to λ-strong-convexity of F_s^λ), but the zero-bias criterion $\mathbb{E}x = x^\star$ is not guaranteed. Nevertheless, an MLMC technique (Algorithm 2) can extract an UnbiasedProx from any ApproxProx. Algorithm 2 repeatedly calls ApproxProx a geometrically-distributed number of times J (every time with x_{init} and x_{prev} equal to the last output), and outputs a point whose expectation equals to an infinite numbers of iterations of ApproxProx, i.e., the exact $\text{prox}_{F,\lambda}(s)$. Moreover, we show that the linear convergence of the procedure implies that the variance of the result remains appropriately bounded (see Proposition 3.4 below). Algorithm 2 is a variation of an estimator by Blanchet & Glynn (2015) that was previously used in a context similar to ours (Asi et al., 2021). However, prior estimators typically have complexity exponential in J, whereas ours are linear in J.

Finally, we define a warm start procedure required by our method.

Definition 3.3 (WarmStart)**.** *Given convex function $F : \mathcal{X} \to \mathbb{R}$, parameter $\lambda > 0$ and diameter bound R, $x_0 = \textsc{WarmStart}_{F,\lambda}(R^2)$ is a procedure that outputs $x_0 \in \mathcal{X}$ such that $\mathbb{E}F(x_0) - \min_{x'\in\mathcal{X}} F(x') \le \lambda R^2$.*

Note that the exact proximal mapping $x = \text{prox}_{F,\lambda}(s)$ satisfies all the requirements above; replacing $\textsc{ApproxProx}_{F,\lambda}$, $\textsc{UnbiasedProx}_{F,\lambda}$, and $\textsc{WarmStart}_{F,\lambda}$ with $\text{prox}_{F,\lambda}$ recovers the exact accelerated proximal method.

3.2 Complexity Bounds

We begin with a complexity bound for implementing UnbiasedProx via Algorithm 2 (proved in Appendix B).

Proposition 3.4 (MLMC turns ApproxProx into UnbiasedProx)**.** *For any convex F and parameter $\lambda > 0$, Algorithm 2 with $p = 1/2$ and $j_0 \ge 2$ implements* UnbiasedProx *and makes $2 + j_0$ calls to* ApproxProx *in expectation.*

We now give our complexity bound for RECAPP and sketch its proof, deferring the full proof to Appendix B.

Theorem 3.5 (RECAPP complexity bound)**.** *Given any convex function $F : \mathcal{X} \to \mathbb{R}$ and parameters $\lambda, R > 0$,* RECAPP *(Algorithm 1) finds $x \in \mathcal{X}$ with $\mathbb{E}F(x) - \min_{x'\in\mathcal{X}} F(x') \le \epsilon$, within $O\big(\sqrt{\lambda R^2/\epsilon}\big)$ iterations using one call to* WarmStart*, and $O\big(\sqrt{\lambda R^2/\epsilon}\big)$ calls to* ApproxProx *and* UnbiasedProx*. If we implement* UnbiasedProx *using Algorithm 2 with $p = 1/2$ and $j_0 = 2$, the total number of calls to* ApproxProx *is $O\big(\sqrt{\lambda R^2/\epsilon}\big)$ in expectation.*

Proof sketch. We split the proof into two steps.

Step 1: Tight idealized potential decrease. Consider iteration t of the algorithm, and define the potential

$$P_t := \mathbb{E}\big[\alpha_t^{-2}\left(F(x_t) - F(x')\right) + \lambda V^{\mathrm{e}}_{v_t}(x')\big],$$

where x' is a minimizer of F in $\mathcal{X}$. Let $x^\star_{t+1} := \mathrm{prox}_{F,\lambda}(s_t)$ and $v^\star_{t+1} = v_t - (\alpha_{t+1})^{-1}(s_t - x^\star_{t+1})$ be the "ideal" values of x_{t+1} and v_{t+1} obtained via an exact prox-point computation, where for simplicity we ignore the projection onto $\mathcal{X}$. Using these points we define the idealized potential

$$P^\star_{t+1} := \mathbb{E}\Big[\alpha_{t+1}^{-2}\left(F(x^\star_{t+1}) - F(x')\right) + \lambda V^{\mathrm{e}}_{v^\star_{t+1}}(x')\Big].$$

Textbook analyses of acceleration schemes show that $P^\star_{t+1} \le P_t$ (Nesterov, 2018; Monteiro & Svaiter, 2013). Basic inexact accelerated prox-point analyses proceed by showing that the true potential is not much worse that the idealized potential, i.e., $P_{t+1} \le P^\star_{t+1} + \delta_t$, which immediately allows one to conclude that $P_T \le P_0 + \Delta_T$ for $\Delta_T = \sum_{t<T}\delta_t$, and therefore that $\mathbb{E}F(x_T) - F(x') \le \alpha_T^2(\lambda R^2 + \Delta_T)$, implying the optimal rate of convergence as long $\Delta_T = O(\lambda R^2)$. However, obtaining such a small Δ_T to be that small naively requires approximating the proximal points to very high accuracy, which is precisely what we attempt to avoid.

Our first step toward a relaxed error criterion is proving stronger idealized potential decrease. We show that,

$$P^\star_{t+1} \le P_t - \mathbb{E}\Big[\alpha_{t+1}^{-2}\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) + \alpha_t^{-2} V^F_{x^\star_{t+1}}(x_t)\Big].$$

While the potential decrease term $\alpha_{t+1}^{-2}\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(s_t)$ is well known and has been thoroughly exploited by prior work (Frostig et al., 2015; Lin et al., 2017; Monteiro & Svaiter, 2013), making use of the term $\alpha_t^{-2} V^F_{x^\star_{t+1}}(x_t)$ is, to the best of our knowledge, new to this work.

Step 2: Matching approximation errors. With the improved potential decrease at bound in hand, our strategy is clear: make the approximation error cancel with the potential decrease. That is, we wish to show

$$P_{t+1} \le P^\star_{t+1} + \mathbb{E}\Big[\alpha_{t+1}^{-2}\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) + \alpha_t^{-2} V^F_{x^\star_{t+1}}(x_t)\Big],$$

so that overall we have $P_{t+1} \le P_t$ and consequently $\mathbb{E}F(x_T) - F(x') = O\big(\alpha_T^2(F(x_0) - F(x') + \lambda R^2\big) = O\big(\lambda R^2/T^2\big)$, with the last bound following form the warm-start condition and $\alpha_t = O(1/t)$.

It remains to show that the ApproxProx and UnbiasedProx criteria provide the needed error bounds. For the function value, Definition 3.1 implies

$$\begin{aligned}\mathbb{E}F(x_{t+1}) - F(s_t) &= \mathbb{E}\Big[F^\lambda_{s_t}(x_{t+1}) - \lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\Big]\\ &\le \mathbb{E}\Bigg[F^\lambda_{s_t}(x^\star_{t+1}) + \frac{\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) + V^F_{x^\star_{t+1}}(x_t)}{8} - \lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\Bigg]\\ &\le F\left(x^\star_{t+1}\right) - F(s_t) + \tfrac{7}{8}\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) + \tfrac{5}{24} V^F_{x^\star_{t+1}}(x_t),\end{aligned}$$

where for the last inequality we use the property that $\mathbb{E}[\frac{1}{2}\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(x_{t+1})] \le \mathbb{E}[\frac{1}{6}\lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)] + \frac{1}{12}V^F_{x^\star_{t+1}}(x_t)$ due to the strong convexity of F^λ and the approximate optimality of x_{t+1} guaranteed by ApproxProx (see (16) and the derivations before it in Appendix).

For $V^{\mathrm{e}}_{v^\star_{t+1}}(x')$, Definition 3.2 implies $v^\star_{t+1} = \mathbb{E}v_{t+1}$ and

$$\begin{aligned}\mathbb{E}V^{\mathrm{e}}_{v_{t+1}}(x') - \mathbb{E}V^{\mathrm{e}}_{v^\star_{t+1}}(x') &= (\alpha_{t+1})^{-2}\tfrac{1}{2}\mathbb{E}\left\|x^\star_{t+1} - \widetilde{x}_{t+1}\right\|^2\\ &\le (\alpha_{t+1})^{-2}\cdot\frac{\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) + V^F_{x^\star_{t+1}}(x_t)}{8\lambda}.\end{aligned}$$

Substituting back into the expressions for P_{t+1} and $P^\star_{t+1}$ and using the fact that $\frac{\alpha_{t+1}^2}{\alpha_t^2} \ge \frac{1}{3}$, we obtain the desired bound on $P_{t+1} - P^\star_{t+1}$ and conclude the proof.[4]

Complexity bound for strongly-convex functions. For completeness, we also include a guarantee for minimizing a strongly-convex function F by restarting RECAPP. See Appendix B for pseudocode and proofs.

Proposition 3.6 (RECAPP for strongly-convex functions)**.** *For any γ-strongly-convex function $F : \mathcal{X} \to \mathbb{R}$, and parameters $\lambda \ge \gamma$, $R > 0$, restarted* RECAPP *(Algorithm 7) finds x such that $\mathbb{E}F(x) - \min_{x'\in\mathcal{X}} F(x') \le \epsilon$, using one call to* WarmStart*, and $O\big(\sqrt{\lambda/\gamma}\log\frac{LR^2}{\epsilon}\big)$ calls to* ApproxProx *and* UnbiasedProx*. If we implement* UnbiasedProx *using Algorithm 2 with $p = 1/2$ and $j_0 = 2$, the number of calls to* ApproxProx *is $O\big(\sqrt{\lambda/\gamma}\log\frac{LR^2}{\epsilon}\big)$ in expectation.*

3.3 Comparisons of Error Criteria

We now compare ApproxProx (Definition 3.1) to other proximal-point error criteria from the literature. Throughout, we fix a center-point s and let $x^\star = \mathrm{prox}_{F,\lambda}(s)$.

Comparison with Frostig et al. (2015). The APPA framework, which focuses on γ-strongly-convex functions, requires the function-value error bound

$$F^\lambda_s(x) - F^\lambda_s(x^\star) \le O\left(\left(\frac{\gamma}{\lambda}\right)^{1.5}\right)\left(F^\lambda_s(x_{\mathrm{init}}) - F^\lambda_s(x^\star)\right)$$

to hold for all x_{init}. To compare this requirement with ApproxProx, note that in the unconstrained setting

$$\begin{aligned}\lambda V^{\mathrm{e}}_{x^\star}(x_{\mathrm{init}}) + V^F_{x^\star}(x_{\mathrm{init}}) &= V^{F^\lambda_s}_{x^\star}(x_{\mathrm{init}})\\ &= F^\lambda_s(x_{\mathrm{init}}) - F^\lambda_s(x^\star),\end{aligned}$$

where the last equality is due to the fact that $x^\star$ minimizes F^λ_s and therefore $\left\langle \nabla F^\lambda_s(x^\star), x^\star - x_{\mathrm{init}}\right\rangle = 0$. Consequently, the error of $\textsc{ApproxProx}_{F,\lambda}(s; x_{\mathrm{init}}, x_{\mathrm{init}})$ is

$$F^\lambda_s(x) - F^\lambda_s(x^\star) \le \frac{1}{8}\left(F^\lambda_s(x_{\mathrm{init}}) - F^\lambda_s(x^\star)\right).$$

[4]The need to have a lower bound like $\frac{\alpha_{t+1}^2}{\alpha_t^2} \ge \frac{1}{3}$ is the reason RECAPP does not take $\alpha_0 = \infty$ and requires a warm-start.

Therefore, in the unconstrained setting and the special case of $x_{\text{prev}} = x_{\text{init}}$ we require a constant factor relative error decrease, while APPA requires decrease by a factor proportional to $(\gamma/\lambda)^{3/2}$, or $(\epsilon/(\lambda R^2))^{3/2}$ with the standard conversion $\gamma = \epsilon/R^2$. Thus, our requirement is significantly more permissive.

Comparison with Lin et al. (2017). The Catalyst framework offers a number of error criteria. Most closely resembling $\textsc{ApproxProx}$ is their relative error criterion (C2):

$$F_s^\lambda(x) - F_s^\lambda(x^\star) \le \delta_t \lambda V_x^{\mathrm{e}}(s),$$

with $\delta_t = (t+1)^{-2}$ which is of the order of $\epsilon/(\lambda R^2)$ for most iterations. Setting $\delta_t = 1/10$ in the Catalyst criterion would satisfy $\textsc{ApproxProx}_{F,\lambda}(s; s, x')$ for any x'. Furthermore, $\textsc{ApproxProx}$ allows for an additional error term proportional to $V_{x^\star}^F(x_{\text{init}})$, which does not exist in Catalyst. In our analysis in the next sections this additional term is essential to efficiently satisfy our criterion.

Comparison with the Monteiro-Svaiter (MS) condition. Ivanova et al. (2021) and Monteiro & Svaiter (2013) consider the error criterion

$$\left\|\nabla F_s^\lambda(x)\right\| \le \sigma\lambda \left\|x - s\right\|.$$

This criterion implies the bound $F_s^\lambda(x) - F_s^\lambda(x^\star) \le \frac{1}{\lambda}\left\|\nabla F_s^\lambda(x)\right\|^2 \le 2\sigma^2\lambda V_x^{\mathrm{e}}(s)$, making it stronger that the Catalyst C2 criterion when $\sigma = \sqrt{\delta_t/2}$. However, Monteiro & Svaiter (2013) show that by updating of v_t using $\nabla F(x_{t+1})$, any *constant* value of σ in $[0,1)$ suffices for obtaining rates similar to those of the exact accelerated proximal point method. The $\textsc{ApproxProx}$ criterion is strictly weaker than the MS criterion with $\sigma = 1/5$.

Ivanova et al. (2021) leverage the MS framework and its improved error tolerance to develop a reduction-based method that, for some problems, is more efficient than Catalyst by a logarithmic factor. However, for the finite sum and max-structured problems we consider in the following sections, it is unclear how to satisfy the MS condition without incurring an extraneous logarithmic complexity term.

Stochastic error criteria. It is important to note that in contrast to APPA and Catalyst, RECAPP is inherently randomized. The unbiased condition of $\textsc{UnbiasedProx}$ is critical to our analysis of the update to v_t. Although in many cases (such as in finite-sum optimization) efficient proximal point oracles require randomization anyhow, we extend the use of randomness to the acceleration framework's update itself. It is an interesting question to determine if this randomization is necessary and comparable performance to RECAPP can be obtained based solely on deterministic applications of $\textsc{ApproxProx}$.

3.4 The AGD Rate as a Special Case

For a quick demonstration of our framework, we show how to recover the classical $\sqrt{LR^2/\epsilon}$ complexity bound for minimizing an L-smooth function F using exact gradient computations. To do so, we set $\lambda = L$ and note that F_s^λ is L-strongly convex and $2L$-smooth. Therefore, for each F_s^λ with $x^\star = \mathrm{prox}_{F,\lambda}(s)$ we can implement $\textsc{ApproxProx}$ by taking 4 gradient steps starting from x_{init}, since these steps produce an x satisfying

$$V_{x^\star}^{\mathrm{e}}(x) \le (1 - \tfrac{1}{2})^4 V_{x^\star}^{\mathrm{e}}(x_{\text{init}}) = \frac{1}{16} V_{x^\star}^{\mathrm{e}}(x_{\text{init}})$$

and therefore

$$F_s^\lambda(x) - F_s^\lambda(x^\star) \le 2L V_{x^\star}^{\mathrm{e}}(x) \le \frac{L}{8} V_{x^\star}^{\mathrm{e}}(x_{\text{init}}).$$

Invoking Theorem 3.5 with $\lambda = L$ shows that RECAPP finds an ϵ-approximate solution with $O(R\sqrt{L/\epsilon})$ gradient queries, recovering the result of Nesterov (1983).

4 Finite-sum Minimization

In this section, we consider the following problem of finite-sum minimization:

$$\underset{x \in \mathcal{X}}{\text{minimize}}\, F(x) := \frac{1}{n}\sum_{i \in [n]} f_i(x), \tag{4}$$

where each f_i is L-smooth and convex.

We solve the problem by combining RECAPP with a single epoch of SVRG (Johnson & Zhang, 2013), shown in Algorithm 3. Our single point of departure from this classical algorithm is that the point we center our gradient estimator at (x_{full}) is allowed to differ from the initial iterate (x_{init}). Setting x_{full} to be the point x_{prev} of $\textsc{ApproxProx}$ allows us to efficiently meet our relaxed error criterion.

Corollary 4.1 ($\textsc{ApproxProx}$ for finite-sum minimization)**.** *Given finite-sum problem (4), points $s, x_{\text{init}}, x_{\text{full}} \in \mathcal{X}$, and $\lambda \in (0, L]$, $\textsc{OneEpochSvrg}$ (Algorithm 3) with $\phi_i(x) := f_i(x) + \frac{\lambda}{2}\left\|x - s\right\|^2$, $\eta \le \frac{1}{32L}$, and $T = \lceil \frac{32}{\eta\lambda} \rceil = O\left(\frac{L}{\lambda}\right)$ implements $\textsc{ApproxProx}_{F,\lambda}(s; x_{\text{init}}, x_{\text{full}})$ (Definition 3.1) using $O(n + L/\lambda)$ gradient queries.*

Corollary 4.1 follows from a slightly more general bound on $\textsc{OneEpochSvrg}$ (Proposition C.1 in Appendix C). Below, we briefly sketch its proof.

Proof sketch for Corollary 4.1. We begin by carefully bounding the variance of the gradient estimator g_t. In Lemma C.3 in the appendix we show that

$$\left\|g_t - \nabla F(x_t)\right\|^2 \le 4L \cdot \left(V_{x^\star}^F(x_{\text{full}}) + V_{x^\star}^{F_s^\lambda}(x_t)\right). \tag{5}$$

Algorithm 3: OneEpochSvrg

Input: $\Phi = \frac{1}{n} \sum_{i \in [n]} \phi_i$ (with component gradient oracles), center point x_{full}, initial point x_{init}, step-size η, iteration number T

Query gradient $\nabla \Phi(x_{\text{full}}) = \frac{1}{n} \sum_{i \in [n]} \nabla \phi_i(x_{\text{full}})$

$x_0 \leftarrow x_{\text{init}}$ ▷ KEY: x_{init}, x_{full} may be different

for $t = 0$ **to** $T - 1$ **do**

  Sample $i_t \sim \text{Unif}[n]$

  $g_t \leftarrow \nabla \phi_{i_t}(x_t) - \nabla \phi_{i_t}(x_{\text{full}}) + \nabla \Phi(x_{\text{full}})$

  $x_{t+1} \leftarrow \mathsf{Proj}_{\mathcal{X}}(x_t - \eta g_t)$

Return: $\bar{x} = \frac{1}{T} \sum_{t \in [T]} x_t$

Algorithm 4: WarmStart-Svrg

Input: $F = \frac{1}{n} \sum_{i \in [n]} f_i$, smoothness L, point x_{init}

Parameter: Iteration number T

$x^{(0)} \leftarrow x_{\text{init}}$

for $k = 0$ ***to*** $K - 1$ **do**

  $\eta_{k+1} \leftarrow \left(8Ln^{2^{-k-1}}\right)^{-1}$

  $x^{(k+1)} \leftarrow$ OneEpochSvrg $\left(F, x^{(k)}, x^{(k)}, \eta_{k+1}, T\right)$

Return: $x^{(K)}$

Next, standard analysis on variance reduced stochastic gradient method (Xiao & Zhang, 2014) shows that

$$\mathbb{E} \left\| x_{t+1} - x^\star \right\|^2 \le \left\| x_t - x^\star \right\|^2 - 2\eta \mathbb{E}\left[F_s^\lambda(x_{t+1}) - F_s^\lambda(x^\star)\right] + \eta^2 \left\| g_t - \nabla F(x_t) \right\|^2.$$

Plugging in the variance bound (5) at iteration x_t, rearranging terms and telescoping for $t \in [T] - 1$, we obtain

$$\begin{aligned}
&\frac{1}{T}\mathbb{E}\sum_{t=1}^{T}\left(F_s^\lambda(x_t) - F_s^\lambda(x^\star)\right) \\
&\le \frac{2}{\eta T} V^{\mathrm{e}}_{x^\star}(x_{\text{init}}) + 4\eta L V^F_{x^\star}(x_{\text{full}}) + \frac{4\eta L}{T} V^{F_s^\lambda}_{x^\star}(x_{\text{init}}) \\
&\overset{(i)}{\le} \frac{1}{8} V^F_{x^\star}(x_{\text{full}}) + \left(\frac{2}{\eta T} + \frac{4\eta L(L+\lambda)}{T}\right) V^{\mathrm{e}}_{x^\star}(x_{\text{init}}) \\
&\overset{(ii)}{\le} \frac{1}{8}\lambda V^{\mathrm{e}}_{x^\star}(x_{\text{init}}) + \frac{1}{8} V^F_{x^\star}(x_{\text{full}}),
\end{aligned}$$

where we use (i) smoothness of F_s^λ, and (ii) the choices of η and T. Noting that $\bar{x} = \frac{1}{T}\sum_{t\in[T]} x_t$ satisfies $F_s^\lambda(\bar{x}) \le \frac{1}{T}\sum_{t=1}^{T}\left(F_s^\lambda(x_t)\right)$ by convexity concludes the proof sketch.

Warm-start implementation. We now explain how to reuse OneEpochSvrg for obtaining a valid warm-start for RECAPP (Definition 3.3). Given any initial iterate with function error Δ, we show that a careful choice of step size for OneEpochSvrg leads to a point with suboptimality $\sqrt{\frac{LR^2\Delta}{n}}$ in $O(n)$ gradient computations (Lemma C.5). Repeating this procedure $O(\log\log n)$ times produces a point with suboptimality $O(LR^2/n)$, which is a valid warm-start for $\lambda = L/n$. We remark that Song et al. (2020) achieve the same $O(n \log\log n)$ complexity with a different procedure that entails changing the recursion for α_t in Line 3 of Algorithm 1. We believe that our approach is conceptually simpler and might be of independent interest.

Corollary 4.2 (WarmStart-Svrg for finite-sum minimization)**.** *Consider problem* (4) *with minimizer $x^\star$, smoothness parameter L, and some initial point x_{init} with $R = \|x_{\text{init}} - x^\star\|$, for any $\lambda \ge L/n$, Algorithm 4 with $T = 32n$, $K = \log\log n$ implements* WarmStart$_{F,\lambda}(R^2)$ *with $O(n\log\log n)$ gradient queries.*

By implementing ApproxProx and WarmStart using OneEpochSvrg, RECAPP provides the following state-of-the-art complexity bound for finite-sum problems.

Theorem 4.3 (RECAPP for finite-sum minimization)**.** *Given a finite-sum problem* (4) *on domain $\mathcal{X}$ with diameter R,* RECAPP *(Algorithm 1) with parameters $\lambda = \frac{L}{n}$ and $T = O(R\sqrt{Ln^{-1}\epsilon^{-1}})$, using* OneEpochSvrg *for* ApproxProx*, and* WarmStart-Svrg *for* WarmStart*, outputs an x such that $\mathbb{E}F(x) - \min_{x' \in \mathcal{X}} F(x') \le \epsilon$. The total gradient query complexity is $O(n\log\log n + \sqrt{nLR^2/\epsilon})$ in expectation. Further, if F is γ-strongly-convex with $\gamma \le O(L/n)$, restarted* RECAPP *(Algorithm 7) finds an ϵ-approximate solution using $O(n\log\log n + \sqrt{nL/\gamma}\log(LR^2/(n\epsilon))$ gradient queries in expectation.*

4.1 Empirical Results

In this section, we provide an empirical performance comparisons between RECAPP and SVRG (Johnson & Zhang, 2013) and Catalyst (Lin et al., 2017). Specifically, we compare to the C1* variant of Catalyst-SVRG, which Lin et al. (2017) report to have the best performance in practice. We implemented all algorithms in Python, using the Numba (Lam et al., 2015) package for just-in-time compilation which significantly improved runtime. Our code is available at: `github.com/yaircarmon/recapp`.

Task and datasets. We consider logistic regression on three datasets from libSVM (lib): covertype ($n = 581,012$, $d = 54$), real-sim ($n = 72,309$, $d = 20,958$), and a9a ($n = 32,561$, $d = 123$). For each dataset we rescale the feature vectors to using unit Euclidean norm so that each f_i is exactly 0.25-smooth. We do not add ℓ_2 regularization to the logistic regression objective.

We defer readers to Appendix C.1 for detailed implementation of the algorithms and parameter tuning.

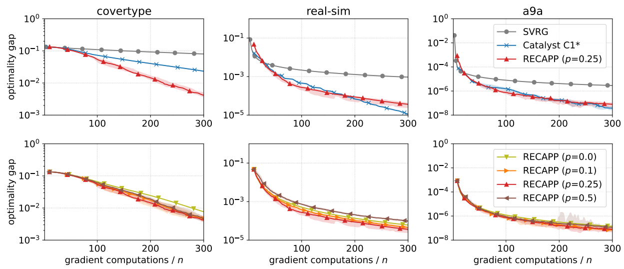

Figure 1: Empirical evaluation of RECAPP on finite sum problems. Columns represent different datasets, the top row compares RECAPP to SVRG and Catalyst, and the bottom row compares different MLMC p parameters ($p = 0$ corresponds to $\tilde{x}_{t+1} = x_{t+1}$ in Algorithm 1, a baseline meant to test whether MLMC is helpful at all, see Appendix C.1 for more details). Solid lines show median over 20 seeds, and shaded regions show interquartile range.

Findings. We summarize our experimental findings in Figure 1. The top row of Figure 1 shows that RECAPP is competitive with Catalyst C1*: on covertype RECAPP is significantly faster, on a9a it is about the same, and on real-sim it is a bit slower. Note that Catalyst C1* incorporates carefully designed heuristics and parameters for choosing the SVRG initialization and stopping time, while the RECAPP implementation directly follows our theoretical development. The bottom row of Figure 1 shows that $p \in \{0.1, 0.25\}$ provides a modest but fairly consistent improvement over the no-MLMC baseline. This provides evidence that MLMC might be beneficial in practice.

5 Max-structured Minimization

In this section, we consider the following problem of the max-structured function minimization:

$$\underset{x\in\mathcal{X}}{\text{minimize}}\, F(x) := \max_{y\in\mathcal{Y}} f(x,y), \tag{6}$$

where $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is L smooth, convex in x (for every y) and μ-strongly-concave in y (for every x).

We solve (6) by combining RECAPP with variants of mirror-prox method (Nemirovski, 2004), shown in Algorithm 5. Given a convex-concave L-smooth objective ϕ, MIRRORPROX (Algorithm 5) starts from initial point $x_{\text{init}}, y_{\text{init}}$, and finds in $O(T)$ gradient queries an approximate solution x_T, y_T satisfying for any $x, y \in \mathcal{X} \times \mathcal{Y}$,

$$\phi(x_T, y) - \phi(x, y_T) \le \frac{L}{T}\left(V^{\text{e}}_{x_{\text{init}}}(x) + V^{\text{e}}_{y_{\text{init}}}(y)\right). \tag{7}$$

Our main observation is that applying such a mirror-prox method to the regularized objective $\phi(x,y) = f(x,y) + \mu V^{\text{e}}_s(x)$, initialized at $(x_{\text{init}}, \mathcal{O}^{\text{br}}_F(x_{\text{prev}}))$ where we define the best response oracle $\mathcal{O}^{\text{br}}_F(x) := \arg\max_{y\in\mathcal{Y}} f(x,y)$, outputs solution satisfying the relaxed error criterion of APPROXPROX$_{F,\mu}$ after $T = O(L/\mu)$ steps. We formalize this observation in Lemma 5.1.

Lemma 5.1 (APPROXPROX for max-structured minimization)**.** *Given max-structured minimization problem (6) and an oracle $\mathcal{O}^{\text{br}}_F(x)$ that outputs $y^{\text{br}}_x := \max_{y\in\mathcal{Y}} f(x,y)$ for any x, MIRRORPROX in Algorithm 5 initialized at $(x_{\text{init}}, \mathcal{O}^{\text{br}}_F(x_{\text{prev}}))$ implements the procedure APPROXPROX$_{F,\mu}(s; x_{\text{init}}, x_{\text{prev}})$ using a total of $O(L/\mu)$ gradient queries and one call to $\mathcal{O}^{\text{br}}_F(\cdot)$.*

Before providing a proof sketch for the lemma, let us remark on the cost of implementing the best response oracle. Since for any fixed x the function $f(x,\cdot)$ is μ-strongly-concave and L-smooth, we can use AGD to find an δ-accurate best response y' to x in $O\left(\sqrt{\frac{L}{\mu}}\log\frac{F(x)-f(x,y')}{\delta}\right)$ gradient queries. Therefore, even for extremely small values of δ we can expect the best-response computation cost to be negligible compared to the $O(L/\mu)$ complexity of the mirror-prox iterations required to implement APPROXPROX.

Proof sketch for Lemma 5.1. We run MIRRORPROX for $T = O(L/\mu)$ steps on $\phi(x,y) = f(x,y) + \mu V^{\text{e}}_s(x)$. By (7) the output (x_T, y_T) satisfies for $x = x^\star = \text{prox}_{F,\lambda}(s)$, $y = y^{\text{br}}_{x_T}$ and arbitrary constant c,

$$\phi(x_T, y^{\text{br}}_{x_T}) - \phi(x^\star, y_T) \le c\mu\left(V^{\text{e}}_{x_{\text{init}}}(x^\star) + V^{\text{e}}_{y_{\text{init}}}(y^{\text{br}}_{x_T})\right).$$

The optimality of $x^\star$ gives $\phi(x^\star, y_T) - \phi(x^\star, y^{\text{br}}_{x^\star}) \le 0$. Combining with the above implies

$$F^\mu_s(x_T) - F^\mu_s(x^\star) \le c\mu\left(V^{\text{e}}_{x_{\text{init}}}(x^\star) + V^{\text{e}}_{y_{\text{init}}}(y^{\text{br}}_{x_T})\right). \tag{8}$$

We now bound the two sides of (8) separately. The strong concavity of ϕ in y allows us to show that $F^\mu_s(x_T) - F^\mu_s(x^\star) \ge \mu V^{\text{e}}_{y^{\text{br}}_{x_T}}(y^\star)$. For the right-hand side, the definition of y_{init} as a best response to x_{prev} yields $\mu V^{\text{e}}_{y_{\text{init}}}(y^\star) \le V^F_{x^\star}(x_{\text{prev}})$. Plugging both inequalities into (8), and choosing sufficiently small c, we see the output satisfies the condition of a APPROXPROX$_{F,\mu}$ oracle, concluding the proof sketch.

Warm-start implementation. We now explain how to apply accelerated gradient descent (AGD) and a recursive use of MIRRORPROX for obtaining a valid warm-start for RECAPP (Definition 3.3).

Lemma 5.2 (WARMSTART for max-structured minimization)**.** *Consider problem (6) where R, R' are diameter bounds for $\mathcal{X}$, $\mathcal{Y}$ respectively. Given initial point $x_{\text{init}}, y_{\text{init}}$, Algorithm 6, with parameters $T = O(L/\mu)$, $K = O(\log(L/\mu))$ and Line 3 implemented using AGD, implements WARMSTART$_{F,\mu}(R^2)$ with*

$$O\left(L/\mu\log(L/\mu) + \sqrt{L/\mu}\log\left(R'/R\right)\right)$$

gradient queries.

Algorithm 5: MIRRORPROX

1 **Input:** Gradient oracle for $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, smoothness L, points $x_{\text{init}}, y_{\text{init}}$, iteration number T
▷ To implement $\textsc{ApproxProx}_{F,\mu}$, we let $y_{\text{init}} \approx \arg\max_{y \in \mathcal{Y}} f(x, y)$
2 **Parameter:** Step-size η
3 Initialize $x_0 \leftarrow x_{\text{init}}$, $y_0 \leftarrow y_{\text{init}}$
4 **for** $t = 0$ **to** $T - 1$ **do**
5     $u_t \leftarrow \arg\min_{x \in \mathcal{X}} \langle \eta \nabla_x \phi(x_t, y_t), x \rangle + V^{\mathrm{e}}_{x_t}(x)$
6     $v_t \leftarrow \arg\min_{y \in \mathcal{Y}} \langle -\eta \nabla_y \phi(x_t, y_t), y \rangle + V^{\mathrm{e}}_{y_t}(y)$
7     $x_{t+1} \leftarrow \arg\min_{x \in \mathcal{X}} \langle \eta \nabla_x \phi(u_t, v_t), x \rangle + V^{\mathrm{e}}_{x_t}(x)$
8     $y_{t+1} \leftarrow \arg\min_{y \in \mathcal{Y}} \langle -\eta \nabla_y \phi(u_t, v_t), y \rangle + V^{\mathrm{e}}_{y_t}(y)$
9 **Return:** x_T, y_T

Algorithm 6: WARMSTART-MIRRORPROX

1 **Input:** Gradient oracle for $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, strong concavity μ, smoothness L, point $(x_{\text{init}}, y_{\text{init}})$
2 **Parameter:** Iteration number T, epoch number K
3 Find y'_{init} so that
$f(x_{\text{init}}, y'_{\text{init}}) - f(x_{\text{init}}, y^{\mathsf{br}}_{x_{\text{init}}}) \le \frac{1}{2} L R^2$
▷ Implemented via AGD
4 Let $\phi := f(x, y) + \mu V^{\mathrm{e}}_{x_{\text{init}}}(x)$ and $L' = L + \mu$
5 Initialize $x^{(0)} \leftarrow x_{\text{init}}$, $y^{(0)} \leftarrow y_{\text{init}}$
6 **for** $k = 0$ **to** $K - 1$ **do**
7     $x^{(k+1)}, y^{(k+1)} \leftarrow$ $\textsc{MirrorProx}(\phi, L', x^{(k)}, y^{(k)}, T)$
8 **Return:** $x^{(K)}$

By implementing APPROXPROX and WARMSTART using MIRRORPROX and WARMSTART-MINIMAX, RECAPP provides the following state-of-the-art complexity bounds for minimizing the max-structured problems.

Theorem 5.3 (RECAPP for minimizing the max-structured problem)**.** *Given $F = \max_{y \in \mathcal{Y}} f(x, y)$ with diameter bounds R, R' on $\mathcal{X}$, $\mathcal{Y}$, respectively,* RECAPP *(Algorithm 1) with parameters $\lambda = \mu$ and $T = O(R\sqrt{\mu/\epsilon})$ and* MIRRORPROX *(Algorithm 5) to implement* APPROXPROX*, and* WARMSTART-MIRRORPROX *(Algorithm 6) to implement* WARMSTART*, outputs a solution x such that $\mathbb{E}\ F(x) - \min_{x' \in \mathcal{X}} F(x') \le \epsilon$. The algorithm uses $O\big(LR/\sqrt{\mu\epsilon} + L/\mu \log(L/\mu) + \sqrt{L/\mu} \log(R'/R)\big)$ gradient queries in expectation and $O\big(R\sqrt{\mu/\epsilon}\big)$ calls to a best-response oracle $\mathcal{O}^{\mathsf{br}}_f(\cdot)$. Further, if F is γ-strongly-convex, restarted* RECAPP *(Algorithm 7) with parameters $\lambda = \mu$, $T = O\left(\sqrt{\mu/\gamma}\right)$, $K = O\left(\log\left(LR^2/\epsilon\right)\right)$ finds an ϵ-approximate solution using $O(LR/\sqrt{\mu\gamma} \log\left(LR^2/\epsilon\right) + L/\mu \log(L/\mu) + \sqrt{L/\mu} \log(R'/R))$ gradient queries in expectation and $O\big(\sqrt{\frac{\mu}{\gamma}} \log \frac{LR^2}{\epsilon}\big)$ calls to $\mathcal{O}^{\mathsf{br}}_f(\cdot)$.*

We remark that for strongly-convex F, the restarted Algorithm 9 not only yields a good approximate solution for F, but also can be transferred to a good approximate primal-dual solution for $f(x, y)$ by taking the best-response to the high-accuracy solution x.

Generalization to the framework. To obtain complexity bounds strictly in terms of gradient queries, we extend the framework of Section 3 to handle small additive errors $\delta \approx \Omega(1/t^4)$ at iteration t when implementing the APPROXPROX procedure as defined in (2). For $x^\star = \arg\min_{x \in \mathcal{X}} F^\lambda_s(x)$ we allow APPROXPROX to return x satisfying

$$\mathbb{E} F^\lambda_s(x) - F^\lambda_s(x^\star) \le \frac{1}{8} \left(\lambda V^{\mathrm{e}}_{x^\star}(x_{\text{init}}) + V^F_{x^\star}(x_{\text{prev}})\right) + \delta. \tag{9}$$

This way, in Lemma 5.1 one can implement the best-response oracle $\mathcal{O}^{\mathsf{br}}_f(\cdot)$ (in Line 3) to a sufficient high accuracy using $\widetilde{O}(\sqrt{L/\mu})$ gradient queries, using the standard accelerated gradient method (Nesterov, 1983). This turns the method in Theorem 5.3 into a complete algorithm for solving (6), and only incurs an additional cost of $\widetilde{O}\left(R\sqrt{L/\mu} \cdot \sqrt{\mu/\epsilon}\right) = \widetilde{O}\left(R\sqrt{L/\epsilon}\right)$ gradient queries.

We state the main result here and refer readers to Appendix E for the generalization of RECAPP (Algorithm 8, Algorithm 9 and Proposition E.3) and more detailed discussion.

Theorem 5.4 (RECAPP for minimizing the max-structured problem, without $\mathcal{O}^{\mathsf{br}}_f$)**.** *Under the same setting of Theorem 5.3, Algorithm 8 with accelerated gradient descent to implement $\mathcal{O}^{\mathsf{br}}_f(\cdot)$, outputs a primal ϵ-approximate solution x and has expected gradient query complexity of $O\left(\frac{LR}{\sqrt{\mu\epsilon}} + \frac{L}{\mu} \log \frac{LR'}{\mu R} + R\sqrt{\frac{L}{\epsilon}} \log \frac{L(R+R')^2}{\epsilon}\right)$. Further, if F is γ-strongly-convex, restarted* RECAPP *(Algorithm 9) finds an ϵ-approximate solution and has expected gradient query complexity of $O\left(\frac{LR}{\sqrt{\mu\gamma}} \log\left(\frac{LR^2}{\epsilon}\right) + \frac{L}{\mu} \log\left(\frac{LR'}{\mu R}\right) + \sqrt{\frac{L}{\gamma}} \log\left(\frac{\mu L(R+R')^2}{\gamma\epsilon}\right) \log\left(\frac{LR^2}{\epsilon}\right)\right)$.*

Acknowledgements

The authors thank anonymous reviewers for helpful suggestions. YC was supported in part by the Israeli Science Foundation (ISF) grant no. 2486/21 and the Len Blavatnik and the Blavatnik Family foundation. YJ was supported in part by a Stanford Graduate Fellowship and the Dantzig-Lieberman Fellowship. AS was supported in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF-1955039, a PayPal research award, and a Sloan Research Fellowship.

References

The LIBSVM data webpage. URL https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv preprint arXiv:1603.05953*, 2016.

Asi, H., Carmon, Y., Jambulapati, A., Jin, Y., and Sidford, A. Stochastic bias-reduced gradient methods. *arXiv preprint arXiv:2106.09481*, 2021.

Azizian, W., Mitliagkas, I., Lacoste-Julien, S., and Gidel, G. A tight and unified analysis of gradient-based methods for a whole spectrum of games. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. *Robust optimization*. Princeton University Press, 2009.

Bertsekas, D. *Nonlinear Programming*. Athena Scientific, 1999.

Blanchet, J. H. and Glynn, P. W. Unbiased Monte Carlo for optimization and functions of expectations via multi-level randomization. In *2015 Winter Simulation Conference (WSC)*, pp. 3656–3667, 2015.

Bubeck, S., Jiang, Q., Lee, Y. T., Li, Y., and Sidford, A. Complexity of highly parallel non-smooth convex optimization. In *Advances in Neural Information Processing Systems*, 2019.

Bullins, B. Highly smooth minimization of non-smooth problems. In *Conference on Learning Theory*, pp. 988–1030, 2020.

Carmon, Y., Jambulapati, A., Jiang, Q., Jin, Y., Lee, Y. T., Sidford, A., and Tian, K. Acceleration with a ball optimization oracle. In *Advances in Neural Information Processing Systems*, 2020.

Carmon, Y., Hausler, D., Jambulapati, A., Jin, Y., and Sidford, A. Optimal and adaptive monteiro-svaiter acceleration. *arXiv:2205.15371*, 2022.

Frostig, R., Ge, R., Kakade, S., and Sidford, A. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, 2015.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Garber, D., Hazan, E., Jin, C., Musco, C., Netrapalli, P., Sidford, A., et al. Faster eigenvector computation via shift-and-invert preconditioning. In *International Conference on Machine Learning (ICML)*, 2016.

Gasnikov, A., Dvurechensky, P., Gorbunov, E., Vorontsova, E., Selikhanovych, D., Uribe, C. A., Jiang, B., Wang, H., Zhang, S., Bubeck, S., Jiang, Q., Lee, Y. T., Li, Y., and Sidford, A. Near optimal methods for minimizing convex functions with Lipschitz p-th derivatives. In *Conference on Learning Theory (COLT)*, 2019.

Giles, M. B. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.

Gower, R. M., Schmidt, M., Bach, F., and Richtárik, P. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

Güler, O. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

Hu, Y., Chen, X., and He, N. On the bias-variance-cost tradeoff of stochastic optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ivanova, A., Pasechnyuk, D., Grishchenko, D., Shulgin, E., Gasnikov, A., and Matyukhin, V. Adaptive catalyst for smooth convex optimization. In *International Conference on Optimization and Applications*, pp. 20–37. Springer, 2021.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems (NeurIPS)*, 2013.

Korpelevich, G. M. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747–756, 1976.

Kovalev, D. and Gasnikov, A. The first optimal algorithm for smooth and strongly-convex-strongly-concave minimax optimization. *arXiv:2205.05653*, 2022.

Lam, S. K., Pitrou, A., and Seibert, S. Numba: A llvm-based python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6, 2015.

Lan, G., Li, Z., and Zhou, Y. A unified variance-reduced accelerated gradient method for convex optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Levy, D., Carmon, Y., Duchi, J. C., and Sidford, A. Large-scale methods for distributionally robust optimization. *Advances in Neural Information Processing Systems*, 2020.

Lin, H., Mairal, J., and Harchaoui, Z. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Lin, H., Mairal, J., and Harchaoui, Z. Catalyst acceleration for first-order convex optimization: from theory to practice. *The Journal of Machine Learning Research*, 18(1): 7854–7907, 2017.

Lin, T., Jin, C., and Jordan, M. I. Near-optimal algorithms for minimax optimization. In *Conference on Learning Theory (COLT)*, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Monteiro, R. D. and Svaiter, B. F. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.

Morgenstern, O. and Von Neumann, J. *Theory of games and economic behavior*. Princeton university press, 1953.

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

Nesterov, Y. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Nesterov, Y. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

Nesterov, Y. *Lectures on convex optimization*, volume 137. Springer, 2018.

Ouyang, Y. and Xu, Y. Lower complexity bounds of first-order methods for convex-concave bilinear saddle-point problems. *Mathematical Programming*, 185(1):1–35, 2021.

Paquette, C., Lin, H., Drusvyatskiy, D., Mairal, J., and Harchaoui, Z. Catalyst acceleration for gradient-based non-convex optimization. *arXiv preprint arXiv:1703.10993*, 2017.

Parikh, N. and Boyd, S. Proximal algorithms. *Foundations and Trends in optimization*, 2014.

Rockafellar, R. T. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Salzo, S. and Villa, S. Inexact and accelerated proximal point algorithms. *Journal of Convex analysis*, 19(4):1167–1192, 2012.

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International conference on machine learning (ICML)*, 2014.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Song, C., Jiang, Y., and Ma, Y. Variance reduction via accelerated dual averaging for finite-sum optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Song, C., Jiang, Y., and Ma, Y. Unified acceleration of high-order algorithms under Hölder continuity and uniform convexity. *SIAM journal of optimization*, 2021.

Thekumparampil, K. K., Jain, P., Netrapalli, P., and Oh, S. Efficient algorithms for smooth minimax optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Woodworth, B. and Srebro, N. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems*, pp. 3646–3654, 2016.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Yang, J., Zhang, S., Kiyavash, N., and He, N. A catalyst framework for minimax optimization. *Advances in Neural Information Processing Systems*, 2020.

Zhao, R. A primal dual smoothing framework for max-structured nonconvex optimization. *arXiv:2003.04375*, 2020.

A Additional Related Work

Beyond the closely related work already described, our paper touches on several lines of literature.

Finite-sum problems. The ubiquity of finite-sum optimization problems in machine learning has led to a very large body of work on developing efficient algorithms for solving them. We refer the reader to Gower et al. (2020) for a broad survey and focus on *accelerated* finite-sum methods, i.e., with a leading order complexity term scaling as $\sqrt{n/\epsilon}$ (or as $\sqrt{n\kappa}$ for strongly-convex problems with condition number κ). Accelerated Proximal Stochastic Dual Coordinate Ascent (Shalev-Shwartz & Zhang, 2014) gave the first such accelerated rate for an important subclass of finite-sum problems. This method was subsequently interpreted as a special case of APPA/Catalyst (Lin et al., 2015; Frostig et al., 2015), which can also accelerate several other finite-sum optimization problems. Since then, research has focused on designing more practical and theoretically efficient accelerated algoirthms by opening the APPA/Catalyst black box. The algorithms Katyusha (Allen-Zhu, 2016), Varag (Lan et al., 2019) and VRADA (Song et al., 2020) offer improved complexity bound at the price of the generality and simplicity of APPA/Catalyst. Our approach matches the best existing guarantee (due to VRADA) without paying this price.

Max-structured problems. Objectives of the form $F(x) = \max_{y\in\mathcal{Y}} f(x, y)$ are very common in machine learning and beyond. Such objectives arise from constraints (via Lagrange multipliers) (Bertsekas, 1999), robustness requirements (Ben-Tal et al., 2009; Ganin et al., 2016; Madry et al., 2018), and game-theoretic considerations (Morgenstern & Von Neumann, 1953; Silver et al., 2017). When f is convex in x and concave in y, the mirror-prox algorithm minimizes F to accuracy epsilon in $O(LRR'/\epsilon)$ gradient evaluations (with respect to both x and y), where R' is the diameter of $\mathcal{Y}$. This rate can be improved when f is μ-strongly-concave in y. For the special bilinear case $f(x, y) = \phi(x) + \langle y, Ax\rangle - \psi(y)$, where ψ a "simple" μ-strongly-convex function, an improved complexity bound of $O(LR/\sqrt{\mu\epsilon})$ has long been known (Nesterov, 2005).

More recent work studies the case of general convex-strongly-concave f. Thekumparampil et al. (2019) and Zhao (2020) establish complexity bounds of $O(\frac{L^{3/2}}{\mu\sqrt{\epsilon}} \log^2 \frac{L^2RR'}{\mu\epsilon})$, which Lin et al. (2020) improve to $O(\frac{L}{\sqrt{\mu\epsilon}} \log^3 \frac{L^2RR'}{\mu\epsilon})$ using an algorithm loosely based on APPA/Catalyst. Yang et al. (2020) present a more direct application of APPA/Catalyst to min-max problems, further improving the complexity to $O(\frac{L}{\sqrt{\mu\epsilon}} \log \frac{L^2R^2}{\mu\epsilon})$, with logarithmic dependence on R' only in a lower order term. Similarly to standard APPA/Catalyst, the min-max variant requires highly accurate proximal point computation, e.g., to function-value error of $O(\frac{\mu^3\epsilon^2}{L^4R^2})$. In contrast, RECAPP requires constant (relative) suboptimality and removes the final logarithmic factor from the leading-order complexity term. Yang et al. (2020) also provide extensions to finite-sum min-max problems and problems where f is non-convex in x, which would likely benefit from out method as well (see Appendix F).

Independent work. In recent independent work, Kovalev & Gasnikov (2022) develop a method that minimizes $\max_y f(x, y)$ assuming μ-strong-concavity in y and γ-strong-convexity in x. They attain an essentially optimal complexity proportional to $\frac{1}{\sqrt{\mu\gamma}}$ times a logarithmic factor depending on problem parameters. Their method is tailored to saddle point problems, working in an expanded space by using point-wise conjugate function and applying recent advances in monotone operator theory. We note that RECAPP with restarts attains the same complexity bound (see Theorem 5.4). However, it is unclear whether the algorithm of (Kovalev & Gasnikov, 2022) can recover the RECAPP's complexity bound in the setting where f is not strongly convex in x.

Monteiro-Svaiter-type acceleration. Monteiro-Svaiter (Monteiro & Svaiter, 2013) propose a variant of the accelerated proximal point method that uses an additional gradient evaluation to facilitate approximate proximal point computation. The Monteiro-Svaiter method and its extensions (Gasnikov et al., 2019; Bubeck et al., 2019; Bullins, 2020; Carmon et al., 2020; Song et al., 2021; Kovalev & Gasnikov, 2022; Carmon et al., 2022) also allow for the regularization parameter λ to be determined dynamically by the procedure approximating the proximal point. Ivanova et al. (2021) leverage this technique to develop a variant of Catalyst that offers improved adaptivity and, in certain cases, improved complexity. We provide additional comparison between the approximation condition of (Monteiro & Svaiter, 2013; Ivanova et al., 2021) and RECAPP in Section 3.3.

Multilevel Monte Carlo (MLMC). MLMC is a method for debiasing function estimators by randomizing over the level of accuracy (Giles, 2015). While originally conceived for PDEs and system simulation, a particular variant of MLMC due to Blanchet & Glynn (2015) has found recent applications in the theory of stochastic optimization (Levy et al., 2020;

Hu et al., 2021). Our method directly builds on the recent proposal of Asi et al. (2021) to use MLMC in order to obtain unbiased estimates of proximal points (or, equivalently, the Moreau envelope gradient). Asi et al. (2021) apply this estimator to de-bias proximal points estimated via SGD and improve several structured acceleration schemes. In contrast, we apply MLMC on linearly convergent algorithms, allowing us to configure it much more aggressively and avoid the extraneous logarithmic factors that appeared in the rates of Asi et al. (2021).

B Proofs for Section 3

We first give a formal proof of Proposition 3.4.

Proposition 3.4 (MLMC turns ApproxProx into UnbiasedProx). *For any convex F and parameter $\lambda > 0$, Algorithm 2 with $p = 1/2$ and $j_0 \geq 2$ implements* UnbiasedProx *and makes $2 + j_0$ calls to* ApproxProx *in expectation.*

Proof of Proposition 3.4. Let $x^\star := \mathrm{prox}_{F,\lambda}(s)$ and $E_0 := \lambda V^{\mathrm{e}}_{x^\star}(x_{\mathrm{init}}) + V^F_{x^\star}(x_{\mathrm{prev}})$. By definition of ApproxProx and the strong λ strong-convexity of F^λ_s we have for $j = 0$,

$$\mathbb{E}\left[\frac{\lambda}{2}\left\|x^{(0)} - x^\star\right\|^2\right] \leq \mathbb{E}F^\lambda_s\left(x^{(0)}\right) - F^\lambda_s\left(x^\star\right) \leq \frac{1}{8}E_0. \tag{10}$$

Further, for all $j \geq 1$,

$$\begin{aligned}\mathbb{E}\left[\frac{\lambda}{2}\left\|x^{(j)} - x^\star\right\|^2\right] &\leq \mathbb{E}F^\lambda_s\left(x^{(j)}\right) - F^\lambda_s\left(x^\star\right) \leq \frac{1}{8}\,\mathbb{E}\left(\lambda V^{\mathrm{e}}_{x^\star}\left(x^{(j-1)}\right) + V^F_{x^\star}(x^{(j-1)})\right)\\ &\overset{(i)}{=} \frac{1}{8}\,\mathbb{E}\left(V^{F^\lambda_s}_{x^\star}(x^{(j-1)})\right) \overset{(ii)}{\leq} \frac{1}{8}\,\mathbb{E}\left(F^\lambda_s\left(x^{(j-1)}\right) - F^\lambda_s\left(x^\star\right)\right) \overset{(iii)}{\leq} \left(\frac{1}{8}\right)^{j+1} E_0,\end{aligned} \tag{11}$$

where we use (i) the equality that $\|a-b\|^2 + \|b-c\|^2 - 2\langle c-b, a-b\rangle = \|a-c\|^2$, (ii), the optimality of $x^\star$ which implies $\langle \nabla F^\lambda_s(x^\star), x - x^\star\rangle \geq 0$ for any $x \in \mathcal{X}$, (iii) induction over j and (10). Consequently, $\mathbb{E}x^{(j)} \to x^\star$ as $j \to \infty$. Further, since $\mathbb{P}[J = k] = p_J$ for all $k \geq j_0$, the algorithm returns a point x satisfying

$$\mathbb{E}x = \mathbb{E}_J\left[x^{(j_0-1)} + p_J^{-1}(x^{(J)} - x^{(J-1)})\right] = \lim_{j\to\infty} x^{(j)} = x^\star,$$

which shows the output is an unbiased estimator of $x^\star$.

Next, to bound the variance, we use that $p_J = 2^{-(J_+ + 1)}$ for $p = 1/2$. Applying (10) and (11) yields that for all $j > j_0$

$$\begin{aligned}\mathbb{E}\left\|x^{(j)} - x^{(j-1)}\right\|^2 &= \mathbb{E}\left\|(x^{(j)} - x^\star) - (x^{(j-1)} - x^\star)\right\|^2 \leq 2\mathbb{E}\left\|x^{(j)} - x^\star\right\|^2 + 2\mathbb{E}\left\|x^{(j-1)} - x^\star\right\|^2\\ &\leq \left(\frac{2}{8} + 2\right)\left(\frac{1}{8}\right)^j\left(\frac{2E_0}{\lambda}\right) = 4.5\cdot\left(\frac{1}{8}\right)^j\left(\frac{E_0}{\lambda}\right).\end{aligned}$$

Consequently,

$$\begin{aligned}\mathbb{E}\left\|p_J^{-1}\left(x^{(J)} - x^{(\max\{J-1,j_0\})}\right)\right\|^2 &= \sum_{j=j_0+1}^{\infty} p_j^{-1}\mathbb{E}\left\|x^{(j)} - x^{(j-1)}\right\|^2 = \sum_{j=1}^{\infty} 2^{j+1}\mathbb{E}\left\|x^{(j_0+j)} - x^{(j_0+j-1)}\right\|^2\\ &\leq 4.5\sum_{j=1}^{\infty}\frac{2^{j+1}}{8^{j_0+j}}\left(\frac{E_0}{\lambda}\right) = \frac{3}{8^{j_0}}\cdot\left(\frac{E_0}{\lambda}\right)\end{aligned}$$

and therefore,

$$\begin{aligned}&\mathbb{E}\left\|x^{(j_0)} + p_J^{-1}\left(x^{(J)} - x^{(J-1)}\right) - x^\star\right\|^2\\ &\leq 2\mathbb{E}\left\|x^{(j_0)} - x^\star\right\|^2 + 2\mathbb{E}\left\|p_J^{-1}\left(x^{(J)} - x^{(\max\{J-1,j_0\})}\right)\right\|^2\\ &\leq \left[4\left(\frac{1}{8}\right)^{j_0+1} + 2\cdot\frac{3}{8^{j_0}}\right]\left(\frac{E_0}{\lambda}\right) = \frac{6.5\cdot E_0}{\lambda\cdot 8^{j_0}}.\end{aligned}$$

Since $j_0 \geq 2$ this implies that the algorithm implements UNBIASEDPROX as claimed.

Finally, note that the expected number of calls made to APPROXPROX is $\mathbb{E}J = j_0 + \mathbb{E}J_+$. Further, $\mathbb{E}J_+ = \frac{1}{1-p}$ since J_+ is geometrically distributed with success probability $1-p$. Consequently, the expected number of calls made to APPROXPROX is $j_0 + \frac{1}{1-p}$ as desired. $\square$

Theorem 3.5 (RECAPP complexity bound)**.** *Given any convex function $F : \mathcal{X} \to \mathbb{R}$ and parameters $\lambda, R > 0$,* RECAPP *(Algorithm 1) finds $x \in \mathcal{X}$ with $\mathbb{E}F(x) - \min_{x' \in \mathcal{X}} F(x') \leq \epsilon$, within $O\left(\sqrt{\lambda R^2/\epsilon}\right)$ iterations using one call to* WARMSTART*, and $O\left(\sqrt{\lambda R^2/\epsilon}\right)$ calls to* APPROXPROX *and* UNBIASEDPROX*. If we implement* UNBIASEDPROX *using Algorithm 2 with $p = 1/2$ and $j_0 = 2$, the total number of calls to* APPROXPROX *is $O\left(\sqrt{\lambda R^2/\epsilon}\right)$ in expectation.*

Notation. We first define the filtration $\mathcal{F}_t = \sigma(x_1, v_1, \ldots, x_t, v_t)$ and use the notation $x_t^\star = \arg\min_{x \in \mathcal{X}} F^\lambda_{s_{t-1}}(x)$ to denote the exact proximal mapping which iteration x_t of the algorithm approximates. We note that $s_t, x^\star_{t+1}, \in \mathcal{F}_t$, i.e., they are deterministic when conditioned on x_t, v_t. We also recall in literature it is well-known that the coefficients α_t we pick satisfy the condition that $\alpha_t \in [\frac{4/\sqrt{3}}{t+3}, \frac{2}{t+2}]$ (Paquette et al., 2017; Yang et al., 2020).

For each iteration of Algorithm 1, we obtain the following bound on potential decrease (a special case and more careful analysis of its variant in Lemma 5 of Asi et al. (2021)).

Proposition B.1. *Under the assumptions of Theorem 3.5, let x' be a minimizer of F. For every t, the iterates of Algorithm 1 satisfy*

$$\mathbb{E}\left[\frac{1}{\alpha_{t+1}^2}(F(x_{t+1}) - F(x')) + \frac{\lambda}{2}\left\|v_{t+1} - x'\right\|^2 \,\middle|\, \mathcal{F}_t\right] \leq \frac{1}{\alpha_t^2}(F(x_t) - F(x')) + \frac{\lambda}{2}\left\|v_t - x'\right\|^2.$$

This proposition can be proved directly by combining the following two lemmas.

Lemma B.2 (Potential decrease guaranteed by exact proximal step)**.** *At t-th iteration of Algorithm 1, let $x^\star_{t+1} := \mathrm{prox}_{F,\lambda}(s_t)$ and $v^\star_{t+1} = v_t - (\alpha_{t+1})^{-1}(s_t - x^\star_{t+1})$ be the "ideal" values of x_{t+1} and v_{t+1} obtained via an exact prox-point computation, then we have*

$$\begin{aligned}&\frac{1}{\alpha_{t+1}^2}\left(F\left(x^\star_{t+1}\right) - F\left(x'\right)\right) + \frac{\lambda}{2}\left\|v^\star_{t+1} - x'\right\|^2 \\ &\quad\leq \frac{1}{\alpha_t^2}(F(x_t) - F(x')) + \frac{\lambda}{2}\left\|v_t - x'\right\|^2 - \frac{\lambda}{\alpha_{t+1}^2} V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) - \frac{1}{\alpha_t^2} V^F_{x^\star_{t+1}}(x_t).\end{aligned} \tag{12}$$

Proof. We let $g^\star_{t+1} = \lambda\left(s_t - x^\star_{t+1}\right)$ and $v^\star_{t+1} = v_t - (\alpha_{t+1})^{-1}(s_t - x^\star_{t+1})$. Now we bound both sides of the quantity $\left\langle g^\star_{t+1}, v_t - x'\right\rangle$. First, note that

$$\frac{1}{\alpha_{t+1}}\left(v_t - x'\right) = \frac{1}{\alpha_{t+1}}\left(x^\star_{t+1} - x'\right) + \frac{1}{\alpha_t^2}\left(x^\star_{t+1} - x_t\right) - \frac{1}{\alpha_{t+1}^2}\left(x^\star_{t+1} - s_t\right).$$

Since $g^\star_{t+1} \in \partial F\left(x^\star_{t+1}\right)$ (see Fact 1.4 by Asi et al. (2021)), F is convex and $\left\langle g^\star_{t+1}, x^\star_{t+1} - s_t\right\rangle = -\lambda\left\|x^\star_{t+1} - s_t\right\|^2$, we have by update of α_t and v_t that

$$\begin{aligned}\frac{1}{\alpha_{t+1}}\left\langle g^\star_{t+1}, v_t - x'\right\rangle &= \frac{1}{\alpha_{t+1}}\left\langle g^\star_{t+1}, x^\star_{t+1} - x'\right\rangle + \frac{1}{\alpha_t^2}\left\langle g^\star_{t+1}, x^\star_{t+1} - x_t\right\rangle - \frac{1}{\alpha_{t+1}^2}\left\langle g^\star_{t+1}, x^\star_{t+1} - s_t\right\rangle \\ &\geq \frac{1}{\alpha_{t+1}}\left(F\left(x^\star_{t+1}\right) - F\left(x'\right)\right) + \frac{1}{\alpha_t^2}\Big(F(x^\star_{t+1}) - F(x_t) + V^F_{x^\star_{t+1}}(x_t)\Big) + \frac{2\lambda}{\alpha_{t+1}^2} V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) \\ &= \frac{1}{\alpha_{t+1}^2}\left(F\left(x^\star_{t+1}\right) - F\left(x'\right)\right) - \frac{1}{\alpha_t^2}(F(x_t) - F(x')) + \frac{2\lambda}{\alpha_{t+1}^2} V^{\mathrm{e}}_{x^\star_{t+1}}(s_t) + \frac{1}{\alpha_t^2} V^F_{x^\star_{t+1}}(x_t).\end{aligned} \tag{13}$$

On the other hand to upper bound $\frac{1}{\alpha_{t+1}}\left\langle g^\star_{t+1}, v_t - x'\right\rangle$, note by definition of $g^\star_{t+1}$ and $v^\star_{t+1}$,

$$\left\|v^\star_{t+1} - x'\right\|^2 = \left\|v_t - \frac{1}{\alpha_{t+1}\lambda} g^\star_{t+1} - x'\right\|^2 = \left\|v_t - x'\right\|^2 - \frac{2}{\alpha_{t+1}\lambda}\left\langle g^\star_{t+1}, v_t - x'\right\rangle + \frac{1}{\alpha_{t+1}^2}\left\|s_t - x^\star_{t+1}\right\|^2.$$

Combining the last two displays and rearranging, we obtain

$$\begin{aligned}
&\frac{1}{\alpha_{t+1}^2}\left(F\left(x_{t+1}^\star\right)-F\left(x'\right)\right)+\frac{\lambda}{2}\left\|v_{t+1}^\star-x'\right\|^2\\
&\le\frac{1}{\alpha_t^2}(F(x_t)-F(x'))+\frac{\lambda}{2}\left\|v_t-x'\right\|^2-\frac{2\lambda}{\alpha_{t+1}^2}V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)-\frac{1}{\alpha_t^2}V^F_{x_{t+1}^\star}(x_t)+\frac{\lambda}{2\alpha_{t+1}^2}\left\|s_t-x_{t+1}^\star\right\|^2\\
&\le\frac{1}{\alpha_t^2}(F(x_t)-F(x'))+\frac{\lambda}{2}\left\|v_t-x'\right\|^2-\frac{\lambda}{\alpha_{t+1}^2}V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)-\frac{1}{\alpha_t^2}V^F_{x_{t+1}^\star}(x_t)\,.
\end{aligned}$$

□

Lemma B.3 (Potential difference between exact and approximate proximal step)**.** *Following the same notation as in Lemma B.2, for x_{t+1} and v_{t+1} defined as in Algorithm 1, we have*

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{\alpha_{t+1}^2}\left(F\left(x_{t+1}\right)-F\left(x'\right)\right)+\frac{\lambda}{2}\left\|v_{t+1}-x'\right\|^2\,\middle|\,\mathcal{F}_t\right]\\
&\quad\le\frac{1}{\alpha_{t+1}^2}\left(F\left(x_{t+1}^\star\right)-F\left(x'\right)\right)+\frac{\lambda}{2}\left\|v_{t+1}^\star-x'\right\|^2+\frac{\lambda}{\alpha_{t+1}^2}V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\frac{1}{\alpha_t^2}V^F_{x_{t+1}^\star}(x_t)\,,
\end{aligned}\tag{14}$$

Proof. To prove (14), we consider the effect of approximation errors of x_{t+1}, v_{t+1} in terms of $x_{t+1}^\star$, $v_{t+1}^\star$, respectively. First, by definition of x_{t+1} and ApproxProx we have that

$$\begin{aligned}
\mathbb{E}[F\left(x_{t+1}\right)\mid\mathcal{F}_t]&=\mathbb{E}\left[F\left(x_{t+1}\right)+\frac{\lambda}{2}\left\|x_{t+1}-s_t\right\|^2\,\middle|\,\mathcal{F}_t\right]-\mathbb{E}\left[\frac{\lambda}{2}\left\|x_{t+1}-s_t\right\|^2\,\middle|\,\mathcal{F}_t\right]\\
&\le F\left(x_{t+1}^\star\right)+\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\frac{1}{8}\left(\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+V^F_{x_{t+1}^\star}(x_t)\right)-\mathbb{E}\left[\lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\,\middle|\,\mathcal{F}_t\right]
\end{aligned}\tag{15}$$

Now we further have

$$\begin{aligned}
\mathbb{E}\left[\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(x_{t+1})\,\middle|\,\mathcal{F}_t\right]&\overset{(i)}{\le}\mathbb{E}[F^\lambda_{s_t}(x_{t+1})\mid\mathcal{F}_t]-F^\lambda_{s_t}(x_{t+1}^\star)\overset{(ii)}{\le}\frac{1}{8}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\frac{1}{8}V^F_{x_{t+1}^\star}(x_t)\\
&\overset{(iii)}{\le}\mathbb{E}\left[\frac{1}{4}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(x_{t+1})+\frac{1}{4}\lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\,\middle|\,\mathcal{F}_t\right]+\frac{1}{8}V^F_{x_{t+1}^\star}(x_t)\,,
\end{aligned}$$

where we once again use (i) the strong convexity of F^λ, (ii) the definition of ApproxProx and (iii) the triangle inequality that $\|a+b\|^2\le 2\|a\|^2+2\|b\|^2$ for any vectors a,b. By rearranging terms and rescaling by a factor of $1/2$ this implies equivalently

$$\mathbb{E}\left[\frac{1}{2}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(x_{t+1})\,\middle|\,\mathcal{F}_t\right]\le\mathbb{E}\left[\frac{1}{6}\lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\,\middle|\,\mathcal{F}_t\right]+\frac{1}{12}V^F_{x_{t+1}^\star}(x_t)\,.\tag{16}$$

Combining the above inequalities we have

$$\begin{aligned}
\mathbb{E}[F\left(x_{t+1}\right)\mid\mathcal{F}_t]&=\mathbb{E}\left[F\left(x_{t+1}\right)+\frac{\lambda}{2}\left\|x_{t+1}-s_t\right\|^2\,\middle|\,\mathcal{F}_t\right]-\mathbb{E}\left[\frac{\lambda}{2}\left\|x_{t+1}-s_t\right\|^2\,\middle|\,\mathcal{F}_t\right]\\
&\overset{(i)}{\le}F\left(x_{t+1}^\star\right)+\frac{7}{8}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\mathbb{E}\left[\frac{1}{2}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(x_{t+1})+\frac{1}{2}\lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\,\middle|\,\mathcal{F}_t\right]+\frac{1}{8}V^F_{x_{t+1}^\star}(x_t)-\mathbb{E}\left[\lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\,\middle|\,\mathcal{F}_t\right]\\
&\overset{(ii)}{\le}F\left(x_{t+1}^\star\right)+\frac{7}{8}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\frac{5}{24}V^F_{x_{t+1}^\star}(x_t)+\mathbb{E}\left[\left(\frac{\lambda}{2}+\frac{\lambda}{6}-\lambda\right)V^{\mathrm{e}}_{x_{t+1}}(s_t)\,\middle|\,\mathcal{F}_t\right]\\
&\le F\left(x_{t+1}^\star\right)+\frac{7}{8}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\frac{5}{24}V^F_{x_{t+1}^\star}(x_t)\,,
\end{aligned}\tag{17}$$

where we use (i) rearranging of terms and using the triangle inequality $\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\frac{1}{8}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)\le\frac{7}{8}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(s_t)+\mathbb{E}\left[\frac{1}{2}\lambda V^{\mathrm{e}}_{x_{t+1}^\star}(x_{t+1})+\frac{1}{2}\lambda V^{\mathrm{e}}_{x_{t+1}}(s_t)\,\middle|\,\mathcal{F}_t\right]$ in (15), and (ii) plugging back the inequality (16).

Now that given definition of UNBIASEDPROX for v_{t+1} so that $\mathbb{E}[v_{t+1} \mid \mathcal{F}_t] = v_{t+1}^\star$ and consequently

$$\begin{aligned}
\mathbb{E}\left[\frac{\lambda}{2}\left\|v_{t+1}-x'\right\|^2 \,\middle|\, \mathcal{F}_t\right] &= \frac{\lambda}{2}\left\|v_{t+1}^\star - x'\right\|^2 + \mathbb{E}\left[\frac{\lambda}{2}\left\|v_{t+1}-v_{t+1}^\star\right\|^2 \,\middle|\, \mathcal{F}_t\right] \\
&\le \frac{\lambda}{2}\left\|v_{t+1}^\star - x'\right\|^2 + \frac{\lambda}{2\alpha_{t+1}^2}\mathbb{E}\left[\left\|\widetilde{x}_{t+1}-x_{t+1}^\star\right\|^2 \,\middle|\, \mathcal{F}_t\right] \\
&\le \frac{\lambda}{2}\left\|v_{t+1}^\star - x'\right\|^2 + \frac{\lambda}{8\alpha_{t+1}^2} V^{\mathrm{e}}_{x_{t+1}^\star}(s_t) + \frac{1}{8\alpha_{t+1}^2} V^F_{x_{t+1}^\star}(x_t).
\end{aligned} \tag{18}$$

Rescaling and summing up inequalities (15) and (18), together with the bound that $(\alpha_t)^2 / (\alpha_{t+1})^2 \le \frac{4(t+4)^2}{16(t+2)^2/3} \le 3$, this proves (14), which also concludes the proof. □

Proof of Theorem 3.5. By requirement of WARMSTART function, we have $F(x_0) - F(x') \le \lambda R^2$. Applying the potential decreasing argument in Proposition B.1 recursively on $t = 0, 1, \cdots, T-1$ thus gives

$$\begin{aligned}
\frac{1}{\alpha_T^2}\mathbb{E}\left[F(x_T) - F(x')\right] &\le \mathbb{E}\left[\frac{1}{\alpha_T^2}\left(F(x_T) - F(x')\right) + \frac{\lambda}{2}\left\|v_T - x'\right\|^2\right] \\
&\le \frac{1}{\alpha_0^2}(F(x_0) - F(x')) + \frac{\lambda}{2}\left\|x_0 - x'\right\|^2 \le \frac{3}{2}\lambda R^2.
\end{aligned}$$

Multiplying both by α_T^2 and using the fact that for $T \ge \lceil\sqrt{6\lambda R^2/\epsilon}\rceil$, $\alpha_T^2 \le \frac{4}{(T+2)^2} \le \frac{2\epsilon}{3\lambda R^2}$, we show that x_T output by Algorithm 1 satisfy that

$$\mathbb{E}\left[F(x_T) - F(x')\right] \le \epsilon.$$

The number of calls to each oracles follow immediately.

When implementing $\text{UNBIASEDPROX}_{F,\lambda}$ using MLMC, guarantees of Proposition 3.4 immediately implies the correctness and the total number of (expected) calls to $\text{APPROXPROX}_{F,\lambda}$. □

We now show an adaptation of our framework to the strongly-convex setting in Algorithm 7. We prove its guarantee as follows.

Algorithm 7: Restarted RECAPP

Input: F: $\mathcal{X} \to \mathbb{R}$, RECAPP
Parameter: $\lambda, R > 0$, iteration number T, epoch number K
Initialize $x^{(0)} \leftarrow \text{WARMSTART}_{F,\lambda}(R^2)$ ▷ To satisfy $\mathbb{E}F(x_0) - \min_{x'} F(x') \le \lambda R^2$
for $k = 0$ **to** $K-1$ **do**
  Run RECAPP on F with $x_0 = v_0 = x^{(k)}$ without WARMSTART (Line 1) for T iterations ▷ Halving error to true optimizer in each iteration and recurse
Return: $x^{(K)}$

Proposition 3.6 (RECAPP for strongly-convex functions)**.** *For any γ-strongly-convex function $F : \mathcal{X} \to \mathbb{R}$, and parameters $\lambda \ge \gamma$, $R > 0$, restarted* RECAPP *(Algorithm 7) finds x such that $\mathbb{E}F(x) - \min_{x' \in \mathcal{X}} F(x') \le \epsilon$, using one call to* WARMSTART*, and $O\big(\sqrt{\lambda/\gamma}\log\frac{LR^2}{\epsilon}\big)$ calls to* APPROXPROX *and* UNBIASEDPROX*. If we implement* UNBIASEDPROX *using Algorithm 2 with $p = 1/2$ and $j_0 = 2$, the number of calls to* APPROXPROX *is $O\big(\sqrt{\lambda/\gamma}\log\frac{LR^2}{\epsilon}\big)$ in expectation.*

Proof of Proposition 3.6. Let x' be the minimizer of F, we show by induction that for the choice of $T = O\left(\sqrt{\lambda/\gamma}\right)$, the iterates $x^{(k)}$ satisfy the condition that

$$\mathbb{E}\left[F\left(x^{(k)}\right) - F(x') + \frac{\lambda}{2}\left\|x^{(k)} - x'\right\|^2\right] \le \frac{3}{2^{k-1}}\lambda R^2, \text{ for } k = 0, 1, \cdots, K. \tag{19}$$

For the base case $k = 0$, we have the inequality holds immediately by definition of procedure WARMSTART. Now suppose the inequality (19) holds for k. For $k+1$, by Proposition B.1 and proof of Theorem 3.5 we obtain

$$\frac{1}{\alpha_T^2}\mathbb{E}\left[F\left(x^{(k+1)}\right) - F\left(x'\right)\right] \le \mathbb{E}\left[F\left(x^{(k)}\right) - F\left(x'\right)\right] + \frac{\lambda}{2}\left\|x^{(k)} - x'\right\|^2 \le \frac{3}{2^{k-1}}\lambda R^2.$$

By our choice of $T = O\left(\sqrt{\lambda/\gamma}\right)$, we have

$$\mathbb{E}\left[F\left(x^{(k+1)}\right) - F\left(x'\right)\right] \le \frac{3}{2\cdot 2^k}\gamma R^2,$$

and consequently by γ-strong convexity it holds that

$$\mathbb{E}\left[\frac{\gamma}{2}\left\|x^{(k+1)} - x'\right\|^2\right] \le \mathbb{E}\left[F\left(x^{(k+1)}\right) - F\left(x'\right)\right] \le \frac{3}{2\cdot 2^k}\gamma R^2 \implies \mathbb{E}\left[\frac{\lambda}{2}\left\|x^{(k+1)} - x'\right\|^2\right] \le \frac{3}{2\cdot 2^k}\lambda R^2.$$

Summing up the two inequalities together we obtain

$$\mathbb{E}\left[F\left(x^{(k+1)}\right) - F(x') + \frac{\lambda}{2}\left\|x^{(k+1)} - x'\right\|^2\right] \le \frac{3}{2^k}\lambda R^2,$$

which shows by math induction that the inequality (19) holds for $k = 0, 1, \cdots, K$.

Now by choice of $K = O\left(\log\left(\lambda R^2/\epsilon\right)\right)$, we have

$$\mathbb{E}\left[F\left(x^{(K)}\right) - F(x')\right] \le \mathbb{E}\left[F\left(x^{(K)}\right) - F(x') + \frac{\lambda}{2}\left\|x^{(K)} - x'\right\|^2\right] \le \frac{3}{2^{K-1}}\lambda R^2 \le \epsilon,$$

which proves the correctness of the algorithm.

The algorithm uses one call to procedure WARMSTART in Line 2. The number of calls to procedures APPROXPROX, UNBIASEDPROX is K times the number of calls within each epoch $k \in [K]$, which is bounded by $O(T)$. The case when implementing UNBIASEDPROX by MLMC and APPROXPROX follows immediately from Proposition 3.4.

□

C Proofs for Section 4

Proposition C.1 (Guarantee for ONEEPOCHSVRG)**.** *For any convex, L-smooth $f_i(x) : \mathcal{X} \to \mathbb{R}$, and parameter $\lambda \ge 0$, consider the finite-sum problem $\Phi(x) := \sum_{i\in[n]} \frac{1}{n}\phi_i(x)$ where $\phi_i(x) := f_i(x) + \frac{\lambda}{2}\|x - s\|^2$. Given a centering point s, an initial point x_{init}, and an anchor point x_{full}, Algorithm 3 with instantiation of $\phi_i(x) = f_i(x) + \frac{\lambda}{2}\|x - s\|^2$, outputs a point $\bar{x} = \frac{1}{T}\sum_{t\in[T]} x_{t-1}$ satisfying*

$$\mathbb{E}F_s^\lambda\left(\bar{x}\right) - F_s^\lambda\left(x^\star\right) \le \frac{2}{\eta T}V^{\mathrm{e}}_{x^\star}\left(x_{\text{init}}\right) + 4\eta L V^F_{x^\star}\left(x_{\text{full}}\right) \;\; \text{where} \;\; x^\star := \arg\min_{x\in\mathcal{X}} \Phi(x)\,.$$

The algorithm uses a total of $O(n+T)$ gradient queries.

To prove Proposition C.1, we first recall the following basic fact for smooth functions.

Lemma C.2. *Let $f : \mathcal{X} \to \mathbb{R}$ be an L-smooth convex function. For any $x, x' \in \mathcal{X}$ we have*

$$\frac{1}{L}\left\|\nabla f(x) - \nabla f(x')\right\|^2 \le \left\langle \nabla f(x) - \nabla f(x'), x - x'\right\rangle \tag{20}$$

and

$$\frac{1}{2L}\left\|\nabla f(x) - \nabla f(x')\right\|^2 \le f(x') - f(x) + \left\langle \nabla f(x), x - x'\right\rangle. \tag{21}$$

We also observe the following few properties of Algorithm 3.

Lemma C.3 (Gradient estimator properties)**.** *For any $x, x_{\rm full} \in \mathcal{X}$, sample $i \in [n]$ uniformly and let $g(x) = \nabla f_i(x) - \nabla f_i(x_{\rm full}) + \nabla F(x_{\rm full}) + \lambda(x-s)$. It holds that $\mathbb{E}\left[g(x)\right] = \nabla F_s^\lambda(x)$, and for $x^\star = \arg\min_{x\in\mathcal{X}} F_s^\lambda(x)$,*

$$\mathbb{E}\left[\left\|g(x) - \nabla F_s^\lambda(x)\right\|^2\right] \le \mathbb{E}\left[\left\|\nabla f_i(x_{\rm full}) - \nabla f_i(x)\right\|^2\right] \le 4L\left(V_{x^\star}^F(x_{\rm full}) + V_{x^\star}^{F_s^\lambda}(x)\right). \tag{22}$$

Proof. First, by definition of how we construct g it holds that

$$\begin{aligned}\mathbb{E}\left[g(x)\right] &= \lambda(x-s) + \nabla F(x_{\rm full}) + \frac{1}{n}\sum_{i=1}^n\left(\nabla f_i(x) - \nabla f_i(x_{\rm full})\right)\\ &= \lambda(x-s) + \nabla F(x_{\rm full}) + \nabla F(x) - \nabla F(x_{\rm full}) = \nabla F_s^\lambda(x).\end{aligned}$$

This proves that g is an unbiased estimator of ∇F_s^λ.

Next, for such unbiased estimator g we have

$$\begin{aligned}\mathbb{E}\left[\left\|g(x) - \nabla F_s^\lambda(x)\right\|^2\right] &= \mathbb{E}\left[\left\|\nabla F(x_{\rm full}) - \nabla F(x) - \nabla f_i(x_{\rm full}) + \nabla f_i(x)\right\|^2\right]\\ &\le \mathbb{E}\left[\left\|\nabla f_i(x_{\rm full}) - \nabla f_i(x)\right\|^2\right],\end{aligned}$$

where we used that $\mathbb{E}\left[\|Z - \mathbb{E}Z\|^2\right] \le \mathbb{E}\left[\|Z\|^2\right]$ for any random Z. This proves the first inequality in (22).

For the second inequality, note

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla f_i(x_{\rm full}) - \nabla f_i(x)\right\|^2\right] &\overset{(i)}{\le} 2\mathbb{E}\left[\left\|\nabla f_i(x_{\rm full}) - \nabla f_i(x^\star)\right\|^2 + \left\|\nabla f_i(x) - \nabla f_i(x^\star)\right\|^2\right]\\ &\overset{(ii)}{\le} 4L\cdot\mathbb{E}\left[f_i(x_{\rm full}) - f_i(x^\star) + \langle\nabla f_i(x^\star), x^\star - x_{\rm full}\rangle + f_i(x) - f_i(x^\star) + \langle\nabla f_i(x^\star), x^\star - x\rangle\right]\\ &= 4L\cdot\left(V_{x^\star}^F(x_{\rm full}) + F(x) - F(x^\star) + \langle\nabla F(x^\star), x^\star - x\rangle\right),\end{aligned} \tag{23}$$

where we use (i) Cauchy-Schwarz inequality for Euclidean norms, and (ii) the property of smoothness of f_i (see Eq. (21)).

Using the standard inequality that $2\langle a-b, b-c\rangle = \|a-c\|^2 - \|b-c\|^2 - \|a-b\|^2$, we also have

$$\begin{aligned}&F(x) - F(x^\star) + \langle\nabla F(x^\star), x^\star - x\rangle\\ &= F_s^\lambda(x) - F_s^\lambda(x^\star) + \langle\nabla F(x^\star) + \lambda(x^\star - s), x^\star - x\rangle - \lambda\langle x^\star - s, x^\star - x\rangle - \frac{\lambda}{2}\|x-s\|^2 + \frac{\lambda}{2}\|x^\star - s\|^2\\ &\le V_{x^\star}^{F_s^\lambda}(x) - \lambda V_{x^\star}^{\rm e}(x) \le V_{x^\star}^{F_s^\lambda}(x).\end{aligned}$$

Substituting this in (23) proves the second inequality in (22). □

The following proof on progress per step follows from the standard analysis by Xiao & Zhang (2014) (as the constrained finite-sum minimization we consider is a special case of theirs), which we include the full statement here for completeness.

Lemma C.4 (Progress per step of OneEpochSvrg, cf. Xiao & Zhang (2014))**.** *Let $x^\star \in \arg\min_{x\in\mathcal{X}}\Phi(x)$ where each ϕ_i is L-smooth and convex. For $\eta \le 1/L$, consider step t in* OneEpochSvrg*, define $\Delta_t = g_t - \nabla\Phi(x_t)$ and $x_{t+1}^\star = \mathsf{Proj}_{\mathcal{X}}(x_t - \eta g_t)$, it holds that*

$$\mathbb{E}\|x_{t+1} - x^\star\|^2 \le \|x_t - x^\star\|^2 - 2\eta\left[\Phi(x_{t+1}) - \Phi(x^\star)\right] + 2\eta^2\mathbb{E}\|\Delta_t\|^2.$$

With these helper lemmas, we are ready to formally prove Proposition C.1.

Proof of Proposition C.1. Consider the t-th step of Algorithm 3, by Lemma C.4 and $\Phi = F_s^\lambda$, for $\eta \le \frac{1}{2L} \le \frac{1}{\lambda+L}$ we have

$$\mathbb{E}\|x_{t+1} - x^\star\|^2 \le \|x_t - x^\star\|^2 - 2\eta\mathbb{E}\left[F_s^\lambda(x_{t+1}) - F_s^\lambda(x^\star)\right] + \eta^2\mathbb{E}\left[\left\|g_t - \nabla F_s^\lambda(x_t)\right\|^2\right].$$

Our bounds on the variance of SVRG plus the L-smoothness of F yields

$$\mathbb{E}\left[\left\|g_t - \nabla F_s^\lambda(x_t)\right\|^2\right] \le 4L\left(V_{x^\star}^F(x_{\text{full}}) + V_{x^\star}^{F_s^\lambda}(x_t)\right).$$

Thus we have by definition of x_{t+1} and divergence,

$$\mathbb{E}V_{x^\star}^{\mathrm{e}}(x_{t+1}) \le V_{x^\star}^{\mathrm{e}}(x_t) - \eta\mathbb{E}\left[F_s^\lambda(x_{t+1}) - F_s^\lambda(x^\star)\right] + 2\eta^2 L V_{x^\star}^{F_s^\lambda}(x_t) + 2\eta^2 L V_{x^\star}^F(x_{\text{full}}). \tag{24}$$

Note $V_{x^\star}^F(x_{\text{full}})$ is independent of x_t. Telescoping bounds (24) and using optimality of $x^\star$ so that $\left\langle \nabla F_s^\lambda(x^\star), x^\star - x\right\rangle \le 0$ for $t \ge 1$, we obtain

$$\mathbb{E}V_{x^\star}^{\mathrm{e}}(x_T) \le V_{x^\star}^{\mathrm{e}}(x_{\text{init}}) - \eta(1-2\eta L)\,\mathbb{E}\left[\sum_{t=0}^{T-1}\left(F_s^\lambda(x_{t+1}) - F_s^\lambda(x^\star)\right)\right] + 2\eta^2 L V_{x^\star}^{F_s^\lambda}(x_{\text{init}}) + 2\eta^2 LTV_{x^\star}^F(x_{\text{full}}).$$

Rearranging terms, dividing over $\eta T/2$, and using convexity of F_s^λ, we have for $\eta \le \frac{1}{32L}$ and $T \ge \frac{32}{\eta\lambda} \ge \frac{128\eta L^2}{\lambda}$

$$\begin{aligned}
\mathbb{E}F_s^\lambda\left(\frac{1}{T}\sum_{t\in[T]} x_t\right) - F_s^\lambda(x^\star) &\le \frac{1}{T}\mathbb{E}\sum_{t=1}^{T}\left(F_s^\lambda(x_t) - F_s^\lambda(x^\star)\right) \le \frac{2(1-2\eta L)}{T}\mathbb{E}\left[\sum_{t=0}^{T-1}\left(F_s^\lambda(x_t) - F_s^\lambda(x^\star)\right)\right]\\
&\le \frac{2}{\eta T}V_{x^\star}^{\mathrm{e}}(x_{\text{init}}) + 4\eta L V_{x^\star}^F(x_{\text{full}}) + \frac{4\eta L}{T}V_{x^\star}^{F_s^\lambda}(x_{\text{init}})\\
&\overset{(\star)}{\le} \frac{1}{8}V_{x^\star}^F(x_{\text{full}}) + \left(\frac{2}{\eta T} + \frac{4\eta L(L+\lambda)}{T}\right)V_{x^\star}^{\mathrm{e}}(x_{\text{init}})\\
&\le \frac{1}{8}V_{x^\star}^F(x_{\text{full}}) + \frac{1}{8}\lambda V_{x^\star}^{\mathrm{e}}(x_{\text{init}}),
\end{aligned}$$

where for inequality $(\star)$ we use the fact that F_s^λ is $(L+\lambda)$-smooth, and the property of smoothness. $\square$

To show how we implement the WARMSTART procedure required in Algorithm 1, we first show the guarantee of the low-accuracy solver for finite-sum minimization of Algorithm 4.

Lemma C.5 (Low-accuracy solver for finite-sum minimization)**.** *For any problem (4) with minimizer $x^\star$, smoothness parameter L, initial point x_{init} so that $R = \|x^\star - x_{\text{init}}\|$, and any $\alpha \ge L/n$, Algorithm 4 with $T = 32n$, finds a point $x^{(K)}$ after K epochs such that $\mathbb{E}\,F\left(x^{(K)}\right) - F(x^\star) \le \frac{1}{2}n^{-1+2^{-K}}LR^2$.*

Proof of Lemma C.5. We prove the argument by math induction and let $c^{(k)} = n^{-1+2^{-k}}$. Note that for the base case $k = 0$, $F(x^{(0)}) - F(x^\star) \le \frac{c^{(0)}}{2}LR^2$ by Eq. (21). Now suppose the above inequality holds for k, i.e. $F(x^{(k)}) - F(x^\star) \le \frac{c^{(k)}}{2}LR^2$. Then for epoch k by guarantee of Proposition C.1 together given choice of $\eta_{k+1} = \frac{1}{8L\sqrt{nc^{(k)}}}$ and $T = 32n$ we have

$$\mathbb{E}F\left(x^{(k+1)}\right) - F(x^\star) \le \frac{2}{\eta_{k+1}T}V_{x^\star}^{\mathrm{e}}\left(x^{(k)}\right) + 4\eta_{k+1}LV_{x^\star}^F\left(x^{(k)}\right) \le \frac{LR^2}{4}\sqrt{\frac{c^{(k)}}{n}} + \frac{LR^2}{4}\sqrt{\frac{c^{(k)}}{n}} = \frac{c^{(k+1)}}{2}LR^2.$$

where for the last inequality we note that series $c^{(k)}$ satisfies $c^{(k+1)} = \sqrt{c^{(k)}/n}$. $\square$

Consequently, after $K = O(\log\log n)$ epochs, we have

$$c^{(K)} \le \frac{2}{n} \implies \mathbb{E}F\left(x^{(K)}\right) - F(x^\star) \le \frac{L}{n}R^2 \le \alpha R^2,$$

for any $\alpha \ge L/n$, which immediately proves the following corollary.

Corollary 4.2 (WARMSTART-SVRG for finite-sum minimization)**.** *Consider problem (4) with minimizer $x^\star$, smoothness parameter L, and some initial point x_{init} with $R = \|x_{\text{init}} - x^\star\|$, for any $\lambda \ge L/n$, Algorithm 4 with $T = 32n$, $K = \log\log n$ implements $\text{WARMSTART}_{F,\lambda}(R^2)$ with $O(n\log\log n)$ gradient queries.*

Now we give the formal proof of Theorem 4.3, the main theorem showing one can use our accelerated scheme RECAPP to solve the finite-sum minimization problem (4) efficiently.

Theorem 4.3 (RECAPP for finite-sum minimization)**.** *Given a finite-sum problem* (4) *on domain* $\mathcal{X}$ *with diameter* R*,* RECAPP *(Algorithm 1) with parameters* $\lambda = \frac{L}{n}$ *and* $T = O(R\sqrt{Ln^{-1}\epsilon^{-1}})$*, using* ONEEPOCHSVRG *for* APPROXPROX*, and* WARMSTART-SVRG *for* WARMSTART*, outputs an* x *such that* $\mathbb{E}F(x) - \min_{x' \in \mathcal{X}} F(x') \le \epsilon$*. The total gradient query complexity is* $O(n \log\log n + \sqrt{nLR^2/\epsilon})$ *in expectation. Further, if* F *is* γ*-strongly-convex with* $\gamma \le O(L/n)$*, restarted* RECAPP *(Algorithm 7) finds an* ϵ*-approximate solution using* $O(n \log\log n + \sqrt{nL/\gamma}\log(LR^2/(n\epsilon))$ *gradient queries in expectation.*

Proof of Theorem 4.3. We first consider the objective function F without strong convexity. The correctness of the algorithm follows directly from Theorem 3.5, together with Corollary 4.1 and Corollary 4.2. For the query complexity, calling WARMSTART-SVRG to implement the procedure of $\text{WARMSTART}_{F,\lambda}(R^2)$ requires gradient queries $O(n \log\log n)$ following Corollary 4.2. The main Algorithm 1 calls $O\left(R\sqrt{\lambda/\epsilon}\right) = O(R\sqrt{Ln^{-1}\epsilon^{-1}})$ of procedure $\text{APPROXPROX}_{F,\lambda}$, which by implementation of ONEEPOCHSVRG each requires $O(n + L/\lambda) = O(n)$ gradient queries following Corollary 4.1. Summing them together gives the claimed gradient complexity in total.

When the objective F is γ-strongly-convex, the proof follows by the same argument as above and the guarantee of restarted RECAPP in Theorem 3.5. □

C.1 Additional Details on Empirical Results

Here we provide additional details for the empirical results in Section 4.1.

SVRG implementation. We implement the SVRG iterates as in Algorithm 3, using $T = 2n$ and $\eta = 4$ (i.e., the inverse of the smoothness of each function). However, instead of outputting the average of all iterates, we return the average of the final $T/2 = n$ iterates.

Catalyst implementation. Our implementation follows closely Catalyst C1* as described in (Lin et al., 2017), where for the subproblem solver we use repeatedly called Algorithm 3 with the parameters and averaging modification described above, checking the C1 termination criterion between each call.

RECAPP implementation. Our RECAPP implementation follows Algorithms 1 and 2, with Algorithm 3 and Algorithm 4 implemented APPROXPROXand WARMSTART, respectively, and Algorithm 3 configured and modified and described above. In Algorithm 2 we set the parameters $j_0 = 0$ and we test $p \in \{0, 0.1, 0.25, 0.5\}$. The setting $p = 0$ which corresponds to setting $\tilde{x}_{t+1} = x_{t+1}$ in Algorithm 1) is a baseline meant to test whether MLMC is helpful at all. For $p > 0$ we change the parameter T in Algorithm 3 such that the *expected* amount of gradient computations is the same as for $p = 0$. Slightly departing from the pseudocode of Algorithm 1, we take x_{t+1} to be $x^{(J)}$ computed in Algorithm 2, rather than $x^{(0)}$, since it is always a more accurate proximal point approximation. We note that our algorithm still has provable guarantees (with perhaps different constant factors) under this configuration.

Parameter tuning. For RECAPP and Catalyst, we tune the proximal regularization parameter λ (called κ in (Lin et al., 2017)). For each problem and each algorithm, we test λ values of the form $\alpha L/n$, where $L = 0.25$ is the objective smoothness, n is the dataset size and α in the set $\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0\}$. We report results for the best λ value for each problem/algorithm pair.

D Proofs for Section 5

We first consider a special case of standard mirror-prox-type methods (Nemirovski, 2004) with Euclidean ℓ_2-divergence on x and y domains separately, i.e. $V_{x,y}(x', y') = V_x^{\mathrm{e}}(x') + V_x^{\mathrm{e}}(y')$. This ensures each step of the methods can be implemented efficiently. Below we state its guarantees, which is standard from literature and we include here for completeness.

Lemma D.1 (T-step guarantee of MIRRORPROX, cf. also Nemirovski (2004))**.** *Let any* $\phi(x, y), (x, y) \in \mathcal{X} \times \mathcal{Y}$ *be a convex-concave, L-smooth function,* $\text{MIRRORPROX}(\phi, L, x_{\text{init}}, y_{\text{init}}, T)$ *in Algorithm 5 with initial points* $(x_{\text{init}}, y_{\text{init}})$ *and* *step size $\eta = 1/L$ outputs a point (x_T, y_T) satisfying for any $x, y \in \mathcal{X} \times \mathcal{Y}$*

$$\phi(x_T, y) - \phi(x, y_T) \le \frac{L}{T}\left(V^{\mathrm{e}}_{x_{\mathrm{init}}}(x) + V^{\mathrm{e}}_{y_{\mathrm{init}}}(y)\right). \tag{25}$$

Next we give complete proofs on the implementation of $\textsc{ApproxProx}_{F,\mu}$ and $\textsc{WarmStart}_{F,\mu}$, using $\textsc{MirrorProx}$ (Algorithm 5) with proper choices of initialization $x_{\mathrm{init}}, y_{\mathrm{init}}$, and $\textsc{WarmStart-Minimax}$ (Algorithm 6), respectively.

Lemma 5.1 ($\textsc{ApproxProx}$ for max-structured minimization)**.** *Given max-structured minimization problem (6) and an oracle $\mathcal{O}^{\mathsf{br}}_F(x)$ that outputs $y^{\mathsf{br}}_x := \max_{y\in\mathcal{Y}} f(x,y)$ for any x, $\textsc{MirrorProx}$ in Algorithm 5 initialized at $(x_{\mathrm{init}}, \mathcal{O}^{\mathsf{br}}_F(x_{\mathrm{prev}}))$ implements the procedure $\textsc{ApproxProx}_{F,\mu}(s; x_{\mathrm{init}}, x_{\mathrm{prev}})$ using a total of $O(L/\mu)$ gradient queries and one call to $\mathcal{O}^{\mathsf{br}}_F(\cdot)$.*

Proof of Lemma 5.1. We incur $\textsc{MirrorProx}$ with $\phi(x,y) = f(x,y) + \frac{\mu}{2}\|x-s\|^2$, smoothness $L+\mu$, initial point $x_{\mathrm{init}}, y_{\mathrm{init}}$, and number of iterations $T = \frac{64(L+\mu)}{\mu}$. By guarantee (25) in Lemma D.1, the algorithm outputs x_T, y_T such that for any $x \in \mathcal{X}, y \in \mathcal{Y}$

$$\phi(x_T, y) - \phi(x, y_T) \le \frac{1}{64}\mu\left(V^{\mathrm{e}}_{x_{\mathrm{init}}}(x) + V^{\mathrm{e}}_{y_{\mathrm{init}}}(y)\right).$$

Suppose ϕ has $(x^\star, y^\star)$ as its unique saddle-point, in particular we pick $x = x^\star$ and $y = y^{\mathsf{br}}_{x_T} := \arg\max_{y\in\mathcal{Y}} f(x_T, y) = \arg\max_{y\in\mathcal{Y}} \phi(x_T, y)$ in the above inequality to obtain

$$\begin{aligned} F^\mu_s(x_T) - F^\mu_s(x^\star) &= \phi\left(x_T, y^{\mathsf{br}}_{x_T}\right) - \phi(x^\star, y^\star) = \left(\phi\left(x_T, y^{\mathsf{br}}_{x_T}\right) - \phi(x^\star, y_T) + (\phi(x^\star, y_T)) - \phi(x^\star, y^\star)\right) \\ &\le \phi\left(x_T, y^{\mathsf{br}}_{x_T}\right) - \phi(x^\star, y_T) \le \frac{1}{64}\mu\left(V^{\mathrm{e}}_{x_{\mathrm{init}}}(x^\star) + V^{\mathrm{e}}_{y_{\mathrm{init}}}\left(y^{\mathsf{br}}_{x_T}\right)\right), \end{aligned} \tag{26}$$

where for the first inequality we use the definition that $y^\star = \arg\max_{y\in\mathcal{Y}} \phi(x^\star, y)$. Now for the LHS of (26), we have

$$\begin{aligned} &F^\mu_s(x_T) - F^\mu_s(x^\star) \\ &\quad\ge \left(1 - \frac{1}{32}\right)\left(F^\mu_s(x_T) - F^\mu_s(x^\star)\right) + \frac{1}{32}\left(\phi\left(x_T, y^{\mathsf{br}}_{x_T}\right) - \phi(x_T, y^\star)\right) + \frac{1}{32}\left(\phi(x_T, y^\star) - \phi(x^\star, y^\star)\right) \\ &\quad\ge \left(1 - \frac{1}{32}\right)\left(F^\mu_s(x_T) - F^\mu_s(x^\star)\right) + \frac{\mu}{32} V^{\mathrm{e}}_{y^{\mathsf{br}}_{x_T}}(y^\star). \end{aligned} \tag{27}$$

Plugging (27) back to (26) and rearranging terms, we obtain

$$\begin{aligned} \left(F^\mu_s(x_T) - F^\mu_s(x^\star)\right) \le& \frac{\mu/64}{1-1/32}\left(V^{\mathrm{e}}_{x_{\mathrm{init}}}(x^\star) + V^{\mathrm{e}}_{y_{\mathrm{init}}}\left(y^{\mathsf{br}}_{x_T}\right)\right) - \frac{\mu/32}{1-1/32} V^{\mathrm{e}}_{y^{\mathsf{br}}_{x_T}}(y^\star) \\ \overset{(\star)}{\le}& \frac{\mu/64}{1-1/32} V^{\mathrm{e}}_{x_{\mathrm{init}}}(x^\star) + \frac{\mu/32}{1-1/32} V^{\mathrm{e}}_{y_{\mathrm{init}}}(y^\star) + \frac{\mu/32}{1-1/32} V^{\mathrm{e}}_{y^{\mathsf{br}}_{x_T}}(y^\star) - \frac{\mu/32}{1-1/32} V^{\mathrm{e}}_{y^{\mathsf{br}}_{x_T}}(y^\star) \\ \le& \frac{1}{16}\mu\left(V^{\mathrm{e}}_{x_{\mathrm{init}}}(x^\star) + V^{\mathrm{e}}_{y_{\mathrm{init}}}(y^\star)\right), \end{aligned} \tag{28}$$

where we use $(\star)$ Cauchy-Schwarz inequality for Euclidean norm.

Further, to bound RHS of (28), we note that by definition of F and $y_{\mathrm{init}} \leftarrow \mathcal{O}^{\mathsf{br}}_f(x_{\mathrm{prev}})$,

$$\begin{aligned} \mu V^{\mathrm{e}}_{y_{\mathrm{init}}}(y^\star) =& \mu V^{\mathrm{e}}_{y^{\mathsf{br}}_{x_{\mathrm{prev}}}}(y^\star) \overset{(i)}{\le} f\left(x_{\mathrm{prev}}, y^{\mathsf{br}}_{x_{\mathrm{prev}}}\right) - f(x_{\mathrm{prev}}, y^\star) \\ \overset{(ii)}{\le}& f\left(x_{\mathrm{prev}}, y^{\mathsf{br}}_{x_{\mathrm{prev}}}\right) - f(x^\star, y^\star) - \langle \nabla_x f(x^\star, y^\star), x_{\mathrm{prev}} - x^\star\rangle = V^F_{x^\star}(x_{\mathrm{prev}}), \end{aligned}$$

where we use (i) strong convexity in y of $-f$ and (ii) convexity of $f(\cdot, y^\star)$.

Plugging this back in (28), we obtain

$$F^\mu_s(x_T) - F^\mu_s(x^\star) \le \frac{\mu}{16} V^{\mathrm{e}}_{x_{\mathrm{init}}}(x^\star) + \frac{1}{8} V^F_{x^\star}(x_{\mathrm{prev}}) \le \frac{1}{8}\left(\mu V^{\mathrm{e}}_{x^\star}(x_{\mathrm{init}}) + V^F_{x^\star}(x_{\mathrm{prev}})\right).$$

Thus we prove that $\textsc{ApproxProx}_{F,\mu}$ can be implemented via $\textsc{MirrorProx}$ properly. The total complexity includes one call to $\mathcal{O}^{\mathsf{br}}_f(\cdot)$ and $O(T) = O(L/\mu)$ gradient queries as each iteration in $\textsc{MirrorProx}$ requires two gradients. $\square$

Lemma 5.2 (WARMSTART for max-structured minimization)**.** *Consider problem* (6) *where* R*,* R' *are diameter bounds for* $\mathcal{X}$*,* $\mathcal{Y}$ *respectively. Given initial point* $x_{\text{init}}, y_{\text{init}}$*, Algorithm 6, with parameters* $T = O(L/\mu)$*,* $K = O(\log(L/\mu))$ *and Line 3 implemented using AGD, implements* $\textsc{WarmStart}_{F,\mu}(R^2)$ *with*

$$O\left(L/\mu \log(L/\mu) + \sqrt{L/\mu}\log\left(R'/R\right)\right)$$

gradient queries.

Proof of Lemma 5.2. Given domain diameter R, R' and the initialization x_{init}, y_{init}, we first use accelerated gradient descent (cf. Nesterov (1983)) to find a $\Theta(LR^2)$-approximate solution of $\max_{y\in\mathcal{Y}} f(x_{\text{init}}, y)$ (which we set to be y'_{init}) using $O\left(\sqrt{L/\mu}\log\left(R'/R\right)\right)$ gradient queries. We recall the definition of $y^{\mathsf{br}}_{x_{\text{init}}} := \arg\max_{y\in\mathcal{Y}} f(x_{\text{init}}, y)$ and thus

$$\mu V^{\mathrm{e}}_{y'_{\text{init}}}\left(y^{\mathsf{br}}_{x_{\text{init}}}\right) \le f(x_{\text{init}}, y'_{\text{init}}) - f(x_{\text{init}}, y^{\mathsf{br}}_{x_{\text{init}}}) \le \frac{1}{2}LR^2. \tag{29}$$

Now we incur MIRRORPROX with objective $\phi(x,y) = f(x,y) + \frac{\mu}{2}\left\|x - x_{\text{init}}\right\|^2$, smoothness $L+\mu$, initial points $(x_{\text{init}}, y'_{\text{init}})$. We let $x^\star(\phi), y^\star$ to denote its unique saddle point. Thus, we have by iterating guarantee of (25) in Lemma D.1 with $T = O(L/\mu)$ iterations, after $K = O(\log L/\mu)$ calls to MIRRORPROX we have

$$\begin{aligned}
V^{\mathrm{e}}_{x^{(K)}}\left(x^\star(\phi)\right) + V^{\mathrm{e}}_{y^{(K)}}\left(x^\star\right) &\le \frac{1}{40}\left(\frac{\mu}{L}\right)^4\left(V^{\mathrm{e}}_{x_{\text{init}}}\left(x^\star(\phi)\right) + V^{\mathrm{e}}_{y'_{\text{init}}}\left(y^\star\right)\right)\\
&\overset{(i)}{\le} \frac{1}{40}\left(\frac{\mu}{L}\right)^4\left(V^{\mathrm{e}}_{x_{\text{init}}}\left(x^\star(\phi)\right) + 2V^{\mathrm{e}}_{y'_{\text{init}}}\left(y^{\mathsf{br}}_{x_{\text{init}}}\right) + 2V^{\mathrm{e}}_{y^{\mathsf{br}}_{x_{\text{init}}}}\left(y^\star\right)\right)\\
&\overset{(ii)}{\le} \frac{1}{40}\left(\frac{\mu}{L}\right)^4\left(\frac{1}{2}R^2 + \frac{LR^2}{\mu} + 2\frac{L^2}{\mu^2}V^{\mathrm{e}}_{x_{\text{init}}}\left(x^\star(\phi)\right)\right)\\
&\le \frac{1}{4}\left(\frac{\mu}{L}\right)^4\frac{L^2}{\mu^2}R^2 = \frac{\mu^2}{4L^2}R^2,
\end{aligned}$$

where we use (i) Cauchy-Schwarz inequality for Euclidean norms, and (ii) condition (29) and the fact that y^{br}_x is L/μ-Lipschitz in x.

Thus given $F(x) = \max_y f(x,y)$ being $(L + L^2/\mu)$-smooth, we have

$$F\left(x^{(K)}\right) - F\left(x^\star(\phi)\right) \le \left(L + L^2/\mu\right)V^{\mathrm{e}}_{x^\star(\phi)}\left(x^{(K)}\right) \le \frac{\mu}{2}R^2. \tag{30}$$

Note we also have

$$F\left(x^\star(\phi)\right) \le F^{\mu}_{x_{\text{init}}}\left(x^\star(\phi)\right) \overset{(\star)}{\le} F^{\mu}_{x_{\text{init}}}\left(x^\star\right) = F\left(x^\star\right) + \frac{\mu}{2}\left\|x^\star - x_{\text{init}}\right\|^2 \le F\left(x^\star\right) + \frac{\mu}{2}R^2,$$

where we use $(\star)$ that $x^\star(\phi)$ minimizes $F^{\mu}_{x_{\text{init}}}(x)$. Plugging this back to (30), we obtain $F\left(x^{(K)}\right) - F\left(x^\star\right) \le \mu R^2$.

The gradient complexity of mirror-prox part is $O(KT) = O(L/\mu\log(L/\mu))$. Summing this together with the gradient complexity for accelerated gradient descent used in obtaining y'_{init} gives the claimed query complexity.

□

E Generalization of Framework and Proof of Theorem 5.4

In this section, we present a generalization of the framework, where we allow additive errors when implementing APPROXPROX and UNBIASEDPROX (Definition E.1 and E.2). When the additive error is small enough, it would contributes to at most $O(\epsilon)$ additive error in the function error and thus generalize our framework (Algorithm 8 and Proposition E.3). In comparison to prior works APPA/Catalyst (Frostig et al., 2015; Lin et al., 2015; 2017), in the application to solving max-structured problems our additive error comes from some efficient method with cheap total gradient costs, thus only contributing to the low-order terms in the oracle complexity (Theorem 5.4).

We first re-define the following procedures of APPROXPROX and UNBIASEDPROX, which also tolerates additive (δ-)error.

Definition E.1 (ApproxProx$^\delta$)**.** *Given convex function $F\colon \mathcal{X} \to \mathbb{R}$, regularization parameter $\lambda > 0$, a centering point $s \in \mathcal{X}$ and two points $x_{\rm init}, x_{\rm prev} \in \mathcal{X}$, $\textsc{ApproxProx}^{\delta}_{F,\lambda}(s; x_{\rm init}, x_{\rm prev})$ is a procedure that outputs an approximate solution x such that for $x^\star = \mathrm{prox}_{F,\lambda}(s) = \arg\min_{x\in\mathcal{X}} F^\lambda_s(x)$,*

$$\mathbb{E}F^\lambda_s(x) - F^\lambda_s(x^\star) \le \frac{1}{8}\left(\lambda V^{\rm e}_{x^\star}(x_{\rm init}) + V^F_{x^\star}(x_{\rm prev})\right) + \delta. \tag{31}$$

Definition E.2 (UnbiasedProx$^\delta$)**.** *Given convex function $F\colon \mathcal{X} \to \mathbb{R}$, regularization parameter $\lambda > 0$, a centering point $s \in \mathcal{X}$, two points $x_{\rm init}, x_{\rm prev} \in \mathcal{X}$, $\textsc{UnbiasedProx}^{\delta}_{F,\lambda}(s; x_{\rm prev})$ is a procedure that outputs an approximate solution x such that $\mathbb{E}\, x = x^\star = \mathrm{prox}_{F,\lambda}(s) = \arg\min_{x\in\mathcal{X}} F^\lambda_s(x)$, and*

$$\mathbb{E}\left\|x - x^\star\right\|^2 \le \frac{1}{4\lambda}\left(\lambda V^{\rm e}_{x^\star}(s) + V^F_{x^\star}(x_{\rm prev})\right) + \frac{2\delta}{\lambda}. \tag{32}$$

Algorithm 8: RECAPP with Additive Error

1 **Input:** $F : \mathcal{X} \to \mathbb{R}$, $\textsc{ApproxProx}^\delta$, $\textsc{UnbiasedProx}^\delta$
2 **Parameter:** $\lambda, R > 0$, iteration number T, $\alpha_0 = 1$
3 Initialize $x_0 \leftarrow \textsc{WarmStart}_{F,\lambda}(R^2)$ ▷ To satisfy $\mathbb{E}\, F(x_0) - F(x^\star) \le \lambda R^2$
4 **for** $t = 0$ **to** $T-1$ **do**
5 Update parameters $\alpha_{t+1} \in [0,1]$ to satisfy $\frac{1}{\alpha_{t+1}^2} - \frac{1}{\alpha_{t+1}} = \frac{1}{\alpha_t}$
6 $s_t \leftarrow (1-\alpha_{t+1})\, x_t + \alpha_{t+1} v_t$
7 $\delta_{t+1} \leftarrow \frac{1}{4t^2}\alpha_t^2 \lambda R^2$ ▷ Additive error decreases as $O(1/t^4)$ to ensure convergence
8 $x_{t+1} \leftarrow \textsc{ApproxProx}^{\delta_{t+1}}_{F,\lambda}(s_t; s_t, x_t)$
9 $\widetilde{x}_{t+1} \leftarrow \textsc{UnbiasedProx}^{\delta_{t+1}}_{F,\lambda}(s_t; x_t)$ ▷ We implement this using $\mathrm{MLMC}^{\delta_{t+1}}(F, \lambda, s_t, x_t)$
10 $v_{t+1} \leftarrow \mathsf{Proj}_{\mathcal{X}}\left(v_t - \frac{1}{\alpha_{t+1}}\left(s_t - \widetilde{x}_{t+1}\right)\right)$
11 **Return:** x_T
12 **function** $\mathrm{MLMC}^\delta(F, \lambda, s, x_{\rm prev})$
13 $\delta_0 \leftarrow 2^{-3}\delta$
14 $x^{(0)} \leftarrow \textsc{ApproxProx}^{\delta_0}_{F,\lambda}(s; s, x_{\rm prev})$
15 Sample random epoch number $J \sim 1 + \mathsf{Geom}\left(\frac{1}{2}\right) \in \{2, 3, \cdots\}$
16 **for** $j = 1$ ***to*** J **do**
17 $\delta_j \leftarrow \frac{1}{4}\delta_{j-1}$
18 $x^{(j)} \leftarrow \textsc{ApproxProx}^{\delta_j}_{F,\lambda}(s; x^{(j-1)}, x^{(j-1)})$
19 **Return:** $x^{(1)} + 2^J(x^{(J)} - x^{(J-1)})$

Algorithm 9: Restarted RECAPP with Additive Error

Input: $F : \mathcal{X} \to \mathbb{R}$, RECAPP with additive error
Parameter: $\lambda, R > 0$, iteration number T, epoch number K, $\alpha_0 = 1$
Initialize $x^{(0)} \leftarrow \textsc{WarmStart}_{F,\lambda}(R^2)$ ▷ To satisfy $\mathbb{E}\, F(x_0) - F(x^\star) \le \lambda R^2$
for $k = 0$ **to** $K-1$ **do**
▷ Halving error to true optimizer and recurse
Run RECAPP (Algorithm 8) on F with $x_0 = v_0 = x^{(k)}$ without WarmStart (Line 3) for T iterations
Return: $x^{(K)}$

With the new definitions of δ-additive proximal oracles and δ-additive unbiased proximal point estimators, we can formally give the guarantee of Algorithm 8 in Proposition E.3.

Proposition E.3 (RECAPP with additive error)**.** *For any convex function $F\colon \mathcal{X} \to \mathbb{R}$, parameters $\lambda, R > 0$,* RECAPP *with additive error (Algorithm 8) finds x such that $\mathbb{E}F(x) - \min_{x'\in\mathcal{X}} F(x') \le \epsilon$ within $O\left(R\sqrt{\lambda/\epsilon}\right)$ iterations. The* *algorithm uses one call to* WARMSTART*, and an expectation of oracle complexity*

$$O\left(\sum_{t\in[T]}\sum_{j=0}^{\infty}\frac{1}{2^j}\mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-2j}\delta_t\right)\right),$$

where we let $\delta_t = \frac{1}{4t^2}\alpha_t^2\lambda R^2 = \Omega(\epsilon^2/(\lambda R^2))$*, and use* $\mathcal{N}(\textsc{ApproxProx}, F, \lambda, \delta)$ *to denote some oracle complexity for calling each* $\textsc{ApproxProx}^{\delta}_{F,\lambda}$.

For any γ*-strongly-convex* $F\colon \mathcal{X} \to \mathbb{R}$*, parameters* $\lambda, R > 0$*, restarted* RECAPP *(Algorithm 9) finds* x *such that* $\mathbb{E}F(x) - \min_{x'\in\mathcal{X}} F(x') \le \epsilon$ *, using one call to* WARMSTART *and an expectation of oracle complexity*

$$O\left(\sum_{k\in[K]}\sum_{t\in[T]}\sum_{j=0}^{\infty}\frac{1}{2^j}\mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-2j}\delta_t^{(k)}\right)\right),$$

where $K = O(\log LR^2/\epsilon)$*,* $T = O\left(\sqrt{\lambda/\gamma}\right)$*, and* $\delta_t^{(k)} = \frac{1}{2^k\cdot 4t^2}\alpha_t^2\lambda R^2 = \Omega\left(\frac{1}{2^k t^4}\lambda R^2\right)$ *for* $t\in[T], k\in[K]$.

To prove the correctness of Proposition E.3, we first show in Lemma E.4 that MLMC^{δ} implements an $\textsc{UnbiasedProx}^{\delta}$ for given F, $\lambda > 0$, with the corresponding inputs. In comparison with the $\delta = 0$ case presented in Section 3, the key difference is we need to ensure when we sample a large index j (with tiny probability), the algorithm calls $\textsc{ApproxProx}^{\delta_j}$ to smaller additive error $\delta_j \approx \Theta(4^{-j})\cdot\delta$, so as to ensure it contributes in total a finite $O(\delta)$ additive term in the variance.

Lemma E.4 (MLMC turns $\textsc{ApproxProx}^{O(\delta)}$ into $\textsc{UnbiasedProx}^{\delta}$)**.** *Given convex* $F\colon \mathcal{X}\to\mathbb{R}$*,* $\lambda > 0$*,* $s, x_{\text{prev}} \in \mathcal{X}$*, function* $\text{MLMC}^{\delta}(F,\lambda,s,x_{\text{prev}})$ *in Algorithm 8 implements* $\textsc{UnbiasedProx}^{\delta}_{F,\lambda}(s;x_{\text{prev}})$*. Denote* $\mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-3}\delta\right)$ *as some oracle complexity for calling each* $\textsc{ApproxProx}^{\delta}_{F,\lambda}$*, then the oracle complexity* $\mathcal{N}$ *for* $\textsc{UnbiasedProx}^{\delta}_{F,\lambda}$ *is* $\mathbb{E}\mathcal{N} = \mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-3}\delta\right) + \mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-5}\delta\right) + \sum_{j=2}^{\infty}\frac{1}{2^{j-2}}\mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-(3+2j)}\delta\right)$.

Proof of Lemma E.4. Let $x^\star = \arg\min_{x\in\mathcal{X}} F_s^\lambda(x)$, by definition of $\textsc{ApproxProx}^{\delta}$, we have

$$\begin{aligned}
\text{for } j=0,\ \mathbb{E}\left[\frac{\lambda}{2}\left\|x^{(0)}-x^\star\right\|^2\right] &\le \mathbb{E}\, F_s^\lambda\left(x^{(0)}\right) - F_s^\lambda\left(x^\star\right) \le \frac{1}{8}\left(\lambda V^{\text{e}}_{x^\star}\left(x_{\text{init}}\right) + V^F_{x^\star}(x_{\text{prev}})\right) + \frac{\delta}{8},\\
\text{for } j\ge 1,\ \mathbb{E}\left[\frac{\lambda}{2}\left\|x^{(j)}-x^\star\right\|^2\right] &\overset{(i)}{\le} \mathbb{E}\, V^{F_s^\lambda}_{x^\star}\left(x^{(j)}\right) \le \mathbb{E}\, F_s^\lambda\left(x^{(j)}\right) - F_s^\lambda\left(x^\star\right)\\
&\le \frac{1}{8}\,\mathbb{E}\left(\lambda V^{\text{e}}_{x^\star}\left(x^{(j-1)}\right) + V^F_{x^\star}(x^{(j-1)})\right) + \frac{\delta}{2\cdot 4^{j+1}}\\
&\overset{(ii)}{\le} \frac{1}{8}\,\mathbb{E}\left(V^{F_s^\lambda}_{x^\star}\left(x^{(j-1)}\right)\right) + \frac{\delta}{2\cdot 4^{j+1}}\\
&\le \frac{1}{8}\,\mathbb{E}\left(F_s^\lambda\left(x^{(j-1)}\right) - F_s^\lambda\left(x^\star\right)\right) + \frac{\delta}{2\cdot 4^{j+1}}\\
&\overset{(iii)}{\le} \left(\frac{1}{8}\right)^{j+1}\left(\lambda V^{\text{e}}_{x^\star}\left(x_{\text{init}}\right) + V^F_{x^\star}(x_{\text{prev}})\right) + \delta\left(\sum_{j'=0}^{j}\frac{1}{2\cdot 8^{j'}\cdot 4^{j+1}}\right),
\end{aligned}$$

where we use (i) the optimality of $x^\star$ which implies $\langle\nabla F(x^\star), x - x^\star\rangle \ge 0$ for any $x\in\mathcal{X}$, (ii) the equality that $\|a-b\|^2 + \|b-c\|^2 - 2\langle c-b, a-b\rangle = \|a-c\|^2$, (iii) the induction over j.

In conclusion, this shows that $\mathbb{E}x^{(j)} \to x^\star$ as $j\to\infty$, and thus by choice of $p_j = 1/2^{j-1}$ for $j\ge 2$, the algorithm returns a point x satisfying

$$\mathbb{E}\, x = \mathbb{E}_J\left[x^{(1)} + 2^J(x^{(J)} - x^{(J-1)})\right] = \lim_{j\to\infty} x^{(j)} = x^\star,$$

which shows the output is an unbiased estimator of $x^\star$.

For the variance, we have by a same calculation as in the proof of Proposition 3.4,

$$\begin{aligned}
\mathbb{E}\left\|x^{(1)}+2^J\left(x^{(J)}-x^{(J-1)}\right)-x^\star\right\|^2 &\le \frac{5}{2}\cdot\mathbb{E}\left[\left\|x^{(1)}-x^\star\right\|^2\right]+\sum_{j=2}^{\infty}\frac{9}{2}\cdot 2^j\cdot\mathbb{E}\left[\left\|x^{(j)}-x^\star\right\|^2\right]\\
&\le \frac{2}{\lambda}\left(\frac{5}{2}\left(\frac{1}{8}\right)^2+\sum_{j=2}^{\infty}\frac{9/8}{2}\left(\frac{1}{4}\right)^j\right)\left(\lambda V^{\mathrm{e}}_{x^\star}\left(x_{\mathrm{init}}\right)+V^F_{x^\star}(x_{\mathrm{prev}})\right)+\frac{5\delta}{16\lambda}+\frac{9\delta}{8\lambda}\sum_{j=2}^{\infty}\frac{5}{4}\cdot\frac{1}{2^j}\\
&\le \frac{1}{4\lambda}\left(\lambda V^{\mathrm{e}}_{x^\star}\left(x_{\mathrm{init}}\right)+V^F_{x^\star}(x_{\mathrm{prev}})\right)+\frac{2\delta}{\lambda},
\end{aligned}$$

which proves the bound as claimed.

The query complexity is in expectation

$$\begin{aligned}
\mathbb{E}\mathcal{N}=\sum_{j=2}^{\infty}\frac{1}{2^{j-1}}\sum_{j'=0}^{j}\mathcal{N}\left(\textsc{ApproxProx},F,\lambda,2^{-(3+2j')}\delta\right)&=\mathcal{N}\left(\textsc{ApproxProx},F,\lambda,2^{-3}\delta\right)\\
+\mathcal{N}\left(\textsc{ApproxProx},F,\lambda,2^{-5}\delta\right)&+\sum_{j=2}^{\infty}\frac{1}{2^{j-2}}\mathcal{N}\left(\textsc{ApproxProx},F,\lambda,2^{-(3+2j)}\delta\right).
\end{aligned}$$

□

This shows that we can implement $\textsc{UnbiasedProx}^\delta$ using $\textsc{ApproxProx}^\delta$, similar to the case without additive error δ, as in Proposition 3.4. Now we are ready to provide a complete proof of Proposition E.3, which shows the correctness and complexity of Algorithm 8.

Proof of Proposition E.3. First of all we recall the notation of filtration $\mathcal{F}_t=\sigma(x_1,v_1,\ldots,x_t,v_t)$, $x_t^\star=\arg\min_{x\in\mathcal{X}}F^\lambda_{s_{t-1}}(x)$, $g^\star_{t+1}=\lambda\left(s_t-x^\star_{t+1}\right)$, $v^\star_{t+1}=v_t-(\alpha_{t+1})^{-1}(s_t-x^\star_{t+1})$ and x' as the minimizer of $F:\mathcal{X}\to\mathbb{R}$ (see Appendix B for more detailed discussion).

The majority of the proof still lies in showing the potential decreasing lemma as in Proposition B.1, while also taking into account the extra additive error δ when implementing oracles $\textsc{ApproxProx}^\delta$ and $\textsc{UnbiasedProx}^\delta$.

Following (12), we recall the inequality that

$$\begin{aligned}
&\frac{1}{\alpha^2_{t+1}}\left(F\left(x^\star_{t+1}\right)-F\left(x'\right)\right)+\frac{\lambda}{2}\left\|v^\star_{t+1}-x'\right\|^2\\
&\quad\le\frac{1}{\alpha_t^2}(F(x_t)-F(x'))+\frac{\lambda}{2}\left\|v_t-x'\right\|^2-\frac{\lambda}{\alpha^2_{t+1}}V^{\mathrm{e}}_{x^\star_{t+1}}\left(s_t\right)-\frac{1}{\alpha_t^2}V^F_{x^\star_{t+1}}\left(x_t\right).
\end{aligned}\tag{33}$$

Thus, by definition of x_{t+1}, δ_{t+1} and $\textsc{ApproxProx}^\delta$ we have that

$$\begin{aligned}
\mathbb{E}[F\left(x_{t+1}\right)\mid\mathcal{F}_t]&\le\mathbb{E}\left[F\left(x_{t+1}\right)+\frac{\lambda}{2}\left\|x_{t+1}-s_t\right\|^2\,\middle|\,\mathcal{F}_t\right]\\
&\le F\left(x^\star_{t+1}\right)+\frac{7}{8}\lambda V^{\mathrm{e}}_{x^\star_{t+1}}\left(s_t\right)+\frac{5}{24}V^F_{x^\star_{t+1}}(x_t)+\delta_{t+1}
\end{aligned}$$

Similarly to (18) and its analysis, we also have by definition of $\textsc{UnbiasedProx}^\delta$ that

$$\begin{aligned}
\mathbb{E}\left[\frac{\lambda}{2}\left\|v_{t+1}-x'\right\|^2\,\middle|\,\mathcal{F}_t\right]&=\left\|v^\star_{t+1}-x'\right\|^2+\mathbb{E}\left[\frac{\lambda}{2}\left\|v_{t+1}-v^\star_{t+1}\right\|^2\,\middle|\,\mathcal{F}_t\right]\\
&\le\frac{\lambda}{2}\left\|v^\star_{t+1}-x'\right\|^2+\frac{\lambda}{2\alpha^2_{t+1}}\mathbb{E}\left[\left\|\widetilde{x}_{t+1}-x^\star_{t+1}\right\|^2\,\middle|\,\mathcal{F}_t\right]\\
&\le\frac{\lambda}{2}\left\|v^\star_{t+1}-x'\right\|^2+\frac{\lambda}{2\alpha^2_{t+1}}\left(\frac{\lambda V^{\mathrm{e}}_{x^\star_{t+1}}(s_t)}{4\lambda}+\frac{V^F_{x^\star_{t+1}}(x_t)}{4\lambda}+\frac{2\delta_{t+1}}{\lambda}\right).
\end{aligned}$$

Plugging these back into (33), we conclude that

$$\frac{1}{\alpha_{t+1}^2}\left(\mathbb{E}[F(x_{t+1}) \mid \mathcal{F}_t] - F(x')\right) + \frac{\lambda}{2}\mathbb{E}\Big[\|v_{t+1} - x'\|^2 \,\Big|\, \mathcal{F}_t\Big] \\ \le \frac{1}{\alpha_t^2}(F(x_t) - F(x')) + \frac{\lambda}{2}\|v_t - x'\|^2 + \frac{2\delta_{t+1}}{\alpha_{t+1}^2}.$$

Recursively applying this bound for $t = 0, 1, \cdots, T-1$ and together with the WARMSTART guarantee we have

$$\frac{1}{\alpha_T^2}\left(\mathbb{E}[F(x_{t+1}) \mid \mathcal{F}_t] - F(x')\right) + \frac{\lambda}{2}\mathbb{E}\Big[\|v_{t+1} - x'\|^2 \,\Big|\, \mathcal{F}_t\Big] \\ \le \frac{1}{\alpha_0^2}(F(x_0) - F(x')) + \frac{\lambda}{2}\|x_0 - x'\|^2 + \sum_{t\in[T]}\frac{2\delta_t}{\alpha_t^2} \\ \implies \frac{1}{\alpha_T^2}\left(\mathbb{E}[F(x_{t+1}) \mid \mathcal{F}_t] - F(x')\right) \le \frac{3}{2}\lambda R^2 + \sum_{t\in[T]}\frac{2\delta_t}{\alpha_t^2} \le 4\lambda R^2 \\ \implies \left(\mathbb{E}[F(x_{t+1}) \mid \mathcal{F}_t] - F(x')\right) \le \epsilon,$$

where we use the choice of $\delta_t = \frac{1}{4t^2}\alpha_t^2\lambda R^2$ and that $\sum_{t\in[T]}\frac{1}{t^2} \le \pi^2/6 \le 2$. This shows the correctness of the algorithm.

The algorithm uses $O(1)$ call to WARMSTART. At each iteration $t+1$, by guarantee of implementing UNBIASEDPROX$^\delta$ using MLMC in Lemma E.4, we have the query complexity with respect to APPROXPROX$^\delta$ is in expectation

$$\mathcal{N}(\textsc{ApproxProx}, F, \lambda, \delta_{t+1}) + \mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-3}\delta_{t+1}\right) + \mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-5}\delta_{t+1}\right) \\ + \sum_{j=2}^{\infty}\frac{1}{2^{j-2}}\mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-(3+2j)}\delta_{t+1}\right) = O\left(\sum_{j=0}^{\infty}\frac{1}{2^j}\mathcal{N}\left(\textsc{ApproxProx}, F, \lambda, 2^{-2j}\delta_{t+1}\right)\right)$$

which implies the total oracle complexity through calling APPROXPROX$^\delta$ by summing over $t = 0, 1, \cdots, T-1$.

The strongly-convex case follows by a similar analysis as in the proof of Proposition 3.6. We show by in duction

$$\mathbb{E}\left[F\left(x^{(k)}\right) - F(x') + \frac{\lambda}{2}\left\|x^{(k)} - x'\right\|^2\right] \le \frac{4}{2^{k-1}}\lambda R^2, \text{ for } k = 0, 1, \cdots, K, \tag{34}$$

taking into account that by choice of $\delta_{t+1}^{(k+1)}$, the contribution of the additive errors is always bounded by $\frac{1}{2^k}\lambda R^2$. This choice also implies the expected oracle complexity due to calling APPROXPROX$^{\delta_t^{(k)}}$ differently at each epoch and iteration. □

The additive errors allowed by this framework are helpful to the task of minimizing the max-structured convex objective $F(x) = \max_{y\in\mathcal{Y}} f(x,y)$. This is because we can then use accelerated gradient descent to solve $\max_y f(x,y)$ for the best-response oracle needed in Line 3 to high accuracy before calling calling Algorithm 5 , and show that MIRRORPROX formally implements a APPROXPROX$^\delta$. The resulting gradient complexity has an extra logarithmic term on δ, but only shows up on a low-order $\widetilde{O}(\sqrt{L/\mu})$ terms.

Corollary E.5 (Implementation of APPROXPROX$^\delta$ for minimizing max-structured function)**.** *Given the minimization of max-structured problem in* (6)*, a centering point s, points x_{init}, x_{prev}, one can use accelerated gradient descent to solve to additive error δ for* (3) *and use Lemma 5.1 to implement the procedure* APPROXPROX$^\delta_{F,\mu}(s; x_{\text{init}}, x_{\text{prev}})$*. It uses a total of* $O\left(L/\mu + \sqrt{L/\mu}\log(L(R')^2/\delta)\right)$ *gradient queries.*

Proof of Corollary E.5. Given the initialization x_{init}, we first use accelerated gradient descent Nesterov (1983) to find a δ-approximate solution of $\max_{y\in\mathcal{Y}} f(x_{\text{prev}}, y)$ (which we set to be y_{init}). We recall the definition of $y^{\mathsf{br}}_{x_{\text{prev}}} := \arg\max_{y\in\mathcal{Y}} f(x_{\text{prev}}, y)$ and thus

$$\mu V^{\text{e}}_{y_{\text{init}}}\left(y^{\mathsf{br}}_{x_{\text{prev}}}\right) \le f(x_{\text{prev}}, y_{\text{init}}) - f(x_{\text{prev}}, y^{\mathsf{br}}_{x_{\text{prev}}}) \le \delta \tag{35}$$

using $O(\sqrt{L/\mu}\log(L(R')^2/\delta))$ gradient queries.

Then, we invoke MirrorProx with $\phi(x,y) = f(x,y) + \frac{\mu}{2}\left\|x-s\right\|^2$, initial point $x_{\text{init}}, y_{\text{init}}$, and number of iterations $T = \frac{64(L+\mu)}{\mu}$. The rest of the proof is essentially the same as in Lemma 5.1, with the only exception that when bounding RHS of (28), we note that by choice of y_{init} and the error bound in (35), it becomes

$$\frac{\mu}{2}V^{\text{e}}_{y_{\text{init}}}(y^\star) \le \mu V^{\text{e}}_{y^{\mathsf{br}}_{x_{\text{prev}}}}(y^\star) + \mu V^{\text{e}}_{y_{\text{init}}}\left(y^{\mathsf{br}}_{x_{\text{prev}}}\right) \le f\left(x_{\text{prev}}, y^{\mathsf{br}}_{x_{\text{prev}}}\right) - f\left(x_{\text{prev}}, y^\star\right) + \delta \le V^F_{x^\star}(x_{\text{prev}}) + \delta.$$

Plugging this new bound with additive error $\Theta(\delta)$ back in (28), we obtain

$$F^\mu_s(x_K) - F^\mu_s(x^\star) \le \frac{\mu}{16}V^{\text{e}}_{x_{\text{init}}}(x^\star) + \frac{1}{8}V^F_{x^\star}(x_{\text{prev}}) + \delta \le \frac{1}{8}\left(\mu V^{\text{e}}_{x^\star}(x_{\text{init}}) + V^F_{x^\star}(x_{\text{prev}})\right) + \delta.$$

Thus the procedure implements $\text{ApproxProx}^\delta_{F,\mu}(s; x_{\text{init}}, x_{\text{prev}})$. The total gradient complexity is the complexity in MirrorProx same as Lemma 5.2 plus the extra complexity in implementing $\mathcal{O}^{\mathsf{br}}_F(\cdot)$ using accelerated gradient descent, which sums up to $O\left(L/\mu + \sqrt{L/\mu}\log\left(L(R')^2/\delta\right)\right)$ as claimed. $\square$

Theorem 5.4 (RECAPP for minimizing the max-structured problem, without $\mathcal{O}^{\mathsf{br}}_f$)**.** *Under the same setting of Theorem 5.3, Algorithm 8 with accelerated gradient descent to implement $\mathcal{O}^{\mathsf{br}}_f(\cdot)$, outputs a primal ϵ-approximate solution x and has expected gradient query complexity of $O\left(\frac{LR}{\sqrt{\mu\epsilon}} + \frac{L}{\mu}\log\frac{LR'}{\mu R} + R\sqrt{\frac{L}{\epsilon}}\log\frac{L(R+R')^2}{\epsilon}\right)$. Further, if F is γ-strongly-convex, restarted* RECAPP *(Algorithm 9) finds an ϵ-approximate solution and has expected gradient query complexity of $O\left(\frac{LR}{\sqrt{\mu\gamma}}\log\left(\frac{LR^2}{\epsilon}\right) + \frac{L}{\mu}\log\left(\frac{LR'}{\mu R}\right) + \sqrt{\frac{L}{\gamma}}\log\left(\frac{\mu L(R+R')^2}{\gamma\epsilon}\right)\log\left(\frac{LR^2}{\epsilon}\right)\right)$.*

Proof of Theorem 5.4. For the non-strongly-convex case, $T = O\left(R\sqrt{\mu/\epsilon}\right)$, the correctness of the algorithm follows directly from the non-strongly-convex case of Proposition E.3, together with Corollary E.5 and Lemma 5.2. For the query complexity, calling WarmStart-Minimax to implement the procedure of WarmStart to μR^2 error requires $O\left(L/\mu\log(L/\mu) + \sqrt{L/\mu}\log\left(R'/R\right)\right)$ gradient queries by Lemma 5.2. Following Proposition E.3, denote $\mathcal{N}\left(\text{ApproxProx}, F, \mu, \delta\right)$ to be the gradient complexity of implementing $\text{ApproxProx}^\delta_{F,\mu}$: we have $\mathcal{N}\left(\text{ApproxProx}, F, \mu, \delta\right) = O\left(L/\mu + \sqrt{L/\mu}\log(L(R')^2/\delta)\right)$ by Corollary E.5. Consequently, the total gradient complexity for implementing all ApproxProx^δ is in expectation

$$\begin{aligned}
O\left(\sum_{t\in[T]}\sum_{j=0}^{\infty}\frac{1}{2^j}\mathcal{N}\left(\text{ApproxProx}, F, \mu, 2^{-2j}\delta_t\right)\right) &= O\left(R\sqrt{\frac{\mu}{\epsilon}}\left(\frac{L}{\mu} + \sqrt{\frac{L}{\mu}}\log\left(\frac{L(R'+R)^2}{\epsilon}\right)\right)\right)\\
&= O\left(\frac{LR}{\sqrt{\mu\epsilon}} + \sqrt{\frac{LR^2}{\epsilon}}\log\left(\frac{L(R'+R)^2}{\epsilon}\right)\right),
\end{aligned}$$

where we use $\delta_t \ge \Omega\left(\frac{1}{T^4}\mu R^2\right) = \Omega(\epsilon/\sqrt{T})$ and choice of $T = O\left(R\sqrt{\mu/\epsilon}\right)$.

Summing the gradient query complexity from both WarmStart and ApproxProx^δ procedures give the final complexity.

For the γ-strongly-convex case, the correctness of the algorithm follows directly from the strongly-convex case of Proposition E.3, together with Corollary E.5 and Lemma 5.2. The query complexity for calling one WarmStart-Minimax remains unchanged. Following Proposition E.3, denote $\mathcal{N}\left(\text{ApproxProx}, F, \mu, \delta\right)$ to be the gradient complexity of implementing $\text{ApproxProx}^\delta_{F,\mu}$: we have $\mathcal{N}\left(\text{ApproxProx}, F, \lambda, \delta\right) = O\left(L/\mu + \sqrt{L/\mu}\log(L(R')^2/\delta)\right)$ by Corollary E.5.

Consequently, the total gradient complexity for implementing all $\textsc{ApproxProx}^{\delta}$ is in expectation

$$
\begin{aligned}
& O\left(\sum_{k\in[K]}\sum_{t\in[T]}\sum_{j=0}^{\infty}\frac{1}{2^j}\mathcal{N}\left(\textsc{ApproxProx}, F, \mu, 2^{-2j}\delta_t^{(k)}\right)\right) \\
&= O\left(\sqrt{\frac{\mu}{\gamma}}\log\left(\frac{LR^2}{\epsilon}\right)\left(\frac{L}{\mu}+\sqrt{\frac{L}{\mu}}\log\left(\frac{\mu L(R'+R)^2}{\epsilon\gamma}\right)\right)\right) \\
&= O\left(\frac{L}{\sqrt{\mu\gamma}}\log\left(\frac{LR^2}{\epsilon}\right)+\sqrt{\frac{L}{\gamma}}\log\left(\frac{\mu L(R'+R)^2}{\epsilon\gamma}\right)\log\left(\frac{LR^2}{\epsilon}\right)\right),
\end{aligned}
$$

where we use $\delta_t^{(k)} \geq \Omega\left(\frac{1}{2^K \cdot T^4}\mu R^2\right)$ and choice of K and T.

□

F Discussion

This paper proposes an improvement of the APPA/Catalyst acceleration framework, providing an efficiently attainable Relaxed Error Criterion for the Accelerated Prox Point method (RECAPP) that eliminates logarithmic complexity terms from previous result while maintaining the elegant black-box structure of APPA/Catalyst.

The main conceptual drawback of our proposed framework (beyond its reliance on randomization) is that efficiently attaining our relaxed error criterion requires a certain degree of problem-specific analysis as well as careful subproblem solver initialization. In contrast, APPA/Catalyst rely on more standard and readily available linear convergence guarantees (which of course also suffice for RECAPP).

Nevertheless, we believe there are many more situations where efficiently meeting the relaxed criterion is possible. These include variance reduction for min-max problems, smooth min-max problems which are (strongly-)concave in y but not convex in x, and problems amenable to coordinate methods. All of these are settings where APPA/Catalyst is effective (Yang et al., 2020; Frostig et al., 2015; Lin et al., 2017) and our approach can likely be provably better.

Moreover, even when proving improved rates is difficult, ApproxProx can still serve as an improved stopping criterion. This motivates further research into practical variants of ApproxProx that depend only on observable quantities (rather than, e.g. the distance to the true proximal point).