2011

On Solving Large-Scale Finite Minimax Problems using Exponential Smoothing

E.Y. Pee

44. E.Y. Pee and J.O. Royset, 2011, "On Solving Large-Scale Finite Minimax Problems using Exponential Smoothing," Journal of Optimization Theory and Applications, Vol. 148, No. 2, pp. 390-421.

On Solving Large-Scale Finite Minimax Problems using Exponential Smoothing[*]

E. Y. Pee[†] and J. O. Royset[‡]

This paper focuses on finite minimax problems with many functions, and their solution by means of exponential smoothing. We conduct run-time complexity and rate of convergence analysis of smoothing algorithms and compare them with those of SQP algorithms. We find that smoothing algorithms may have only sublinear rate of convergence, but as shown by our complexity results, their slow rate of convergence may be compensated by small computational work per iteration. We present two smoothing algorithms with active-set strategies and novel precision-parameter adjustment schemes. Numerical results indicate that the algorithms are competitive with other algorithms from the literature, and especially so when a large number of functions are nearly active at stationary points.

Key Words. Finite minimax, exponential smoothing technique, rate of convergence, run-time complexity.

1 Introduction

There are many applications that can be expressed as finite minimax problems of the form

$$(P) \quad \min_{x \in \mathbb{R}^d} \psi(x), \tag{1}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is defined by

$$\psi(x) \overset{\triangle}{=} \max_{j \in Q} f^j(x), \tag{2}$$

and $f^j : \mathbb{R}^d \to \mathbb{R}$, $j \in Q \overset{\triangle}{=} \{1, 2, ..., q\}$, $q \in \mathbb{N} \overset{\triangle}{=} \{1, 2, ...\}$, are smooth functions. Minimax problems of the form (P) may occur in engineering design [1], control system design [2], portfolio optimization [3], best polynomial approximation [4], or as subproblems in semi-infinite minimax algorithms [5]. In this paper, we focus on minimax problems with many func-

[*]The second author acknowledges support from AFOSR Young Investigator grant F1ATA08337G003.

[†]PhD Candidate, Operations Research Department, Naval Postgraduate School, Monterey, California, USA

[‡]Assistant Professor, Operations Research Department, Naval Postgraduate School, Monterey, California, USA, email: joroyset@nps.edu

tions, i.e., large q, which may result from finely discretized semi-infinite minimax problems or optimal control problems.

The non-differentiability of the objective function in (P) poses the main challenge for solving minimax problems, as the standard unconstrained optimization algorithms cannot be applied directly. Many algorithms have been proposed to solve (P); see for example [6–8] and references therein. One approach is sequential quadratic programming (SQP), where (P) is first transcribed into the standard nonlinear constrained problem

$$(P') \quad \min_{(x,z)\in\mathbb{R}^{d+1}} \{z \mid f^j(x) - z \leq 0 \ \ \forall j \in Q\} \tag{3}$$

and then a SQP algorithm is applied to (P'), advantageously exploiting the special structure in the transcribed problem [6, 9]. Other approaches also based on (P') include interior point methods [8, 10, 11] and conjugate gradient methods in conjunction with exact penalties and smoothing [12].

The SQP algorithm in [6], which solves two quadratic programs (QPs) per iteration, appears especially promising for problems with many sequentially related functions, as in the case of finely discretized semi-infinite minimax problems, due to its aggressive active-set strategy. Recently, a SQP algorithm was proposed in [9] requiring to solve only one QP per iteration, while retaining global convergence and superlinear rate of convergence as in [6]. There is no active-set strategy in [9].

In general, an active-set strategy only considers functions that are nearly active at the current iterate, and thus greatly reduces the number of function and gradient evaluations at each iteration of an algorithm. While the number of iterations to solve a problem to required precision may increase, the overall effect may be a reduction in the number of function and gradient evaluations. For example, [13] reports a 75% reduction in the number of gradient evaluations. Reductions in computing time is also reported for active-set strategies in [6].

In smoothing algorithms (see for example [7, 12–15]), the exponential penalty function [16] is used to produce a smooth approximation of $\psi(\cdot)$. Since the problem remains unconstrained, one can use any standard unconstrained optimization algorithm to solve the smoothed problem such as the Armijo Gradient or Newton methods [7] and Quasi-Newton method [13].

A fundamental challenge for smoothing algorithms is that the smoothed problem becomes increasingly ill-conditioned as the approximation gets more accurate. Consequently, the use of smoothing techniques is complicated by the need to balance accuracy of approximation and problem ill-conditioning. The simple static scheme of constructing a single smoothed problem and solving it is highly sensitive to the choice of accuracy and has poor numerical performance [7]. An attempt to address this challenge by using a sequence of smoothed problems was first made in [15], where a precision parameter for the smooth approximation is initially set to a pre-selected value and then increased by a factor of two each iteration. Effectively, the algorithm approximately solves a sequence of gradually more accurate approximations. This open-loop scheme to precision adjustment is sensitive to the multiplication factor [7].

In [7], the authors propose an adaptive precision-parameter adjustment scheme to ensure that the smoothing precision parameter is kept small (and thus controlling the ill-conditioning) when far from a stationary solution, and is increased as a stationary solution is approached. The numerical results show that the scheme produces a better management of ill-conditioning than with static and open-loop schemes. The smoothing algorithms in [15] and [7] do not incorporate any active-set strategy.

Using the adaptive precision-parameter adjustment scheme in [7], [13] presents an active-set strategy for smoothing algorithms that tackles (P) with large q. We note that the convergence result in Theorem 3.3 of [13] may be slightly incorrect as it claims stationarity for all accumulation points of a sequence constructed by their algorithm. However, their proof relies on [7], which guarantees stationarity for only one accumulation point.

This paper examines smoothing algorithms for (P) with large q from two angles. First, we discuss run-time complexity and rate of convergence for such algorithms. While complexity and rate of convergence have been studied extensively for nonlinear programs and minimax problems (see for example [17–23]), the topics have been largely ignored in the specific context of smoothing algorithms for (P). A challenge here is the increasing ill-conditioning of the smoothed problem as the smoothing precision improves. We quantify the degree of ill-conditioning and use the result to analyze complexity and rate of convergence. We find that the rate of convergence may be sublinear, but low computational effort per iteration yields a competitive run-time complexity in terms of q.

Second, we consider implementation and numerical performance of smoothing algorithms. A challenge here is to construct schemes for selecting the precision parameter that guarantee convergence to stationary points and perform well numerically. As discussed above, static and open-loop precision-parameter adjustment schemes result in poor numerical performance and, thus, we develop two adaptive schemes. In extensive tests against other algorithms, smoothing algorithms with the adaptive schemes are competitive, and especially so when a large number of functions are nearly active at stationary points.

The next section describes the exponential smoothing technique. Section 3 defines a smoothing algorithm and discusses its run-time complexity and rate of convergence. Section 4 presents two adaptive precision-parameter adjustment schemes, the resulting smoothing algorithms, and their proofs of convergence. Section 5 contains numerical results.

2 Exponential Smoothing

In this section, we describe the exponential smoothing technique and include for completeness some known results that will be used in later sections.

For ease of analysis of active-set strategies, we consider the problem

$$(P_\Omega) \quad \min_{x\in\mathbb{R}^d} \psi_\Omega(x), \tag{4}$$

where $\psi_\Omega(x) \overset{\triangle}{=} \max_{j\in\Omega} f^j(x)$, and $\Omega \subseteq Q$. When $\Omega = Q$, (P_Q) is identical to (P). For simplicity of notation, we drop subscripts Q in several contexts below. Next, for any $p > 0$ and $\Omega \subseteq Q$, we define a smooth approximating problem to (P_Ω) by

$$(P_{p\Omega}) \quad \min_{x\in\mathbb{R}^d} \psi_{p\Omega}(x), \tag{5}$$

which we refer to as the smoothed problem, where

$$\psi_{p\Omega}(x) \overset{\triangle}{=} \frac{1}{p}\log\left(\sum_{j\in\Omega}\exp\left(pf^j(x)\right)\right) = \psi_\Omega(x) + \frac{1}{p}\log\left(\sum_{j\in\Omega}\exp\left(p(f^j(x)-\psi_\Omega(x))\right)\right) \tag{6}$$

is the exponential penalty function. We denote (P_{pQ}) by (P_p) for brevity. This smoothing technique was introduced in [16] and used in [7, 12–15].

We denote the set of active functions at $x \in \mathbb{R}^d$ by $\widehat{\Omega}(x) \overset{\triangle}{=} \{j \in \Omega | f^j(x) = \psi_\Omega(x)\}$. Except as stated in Appendix B, we denote components of a vector by superscripts.

The parameter $p > 0$ is the smoothing precision parameter, where a larger p implies higher precision as formalized by the following proposition; see for example [13].

Proposition 2.1. *Suppose that $\Omega \subseteq Q$ and $p > 0$.*

(i) If the functions $f^j(\cdot)$, $j \in \Omega$, are continuous, then $\psi_{p\Omega}(\cdot)$ is continuous, and for any $x \in \mathbb{R}^d$, $\psi_{p\Omega}(x)$ decreases monotonically as p increases.

(ii) For any $x \in \mathbb{R}^d$,

$$0 \leq \frac{\log |\widehat{\Omega}(x)|}{p} \leq \psi_{p\Omega}(x) - \psi_\Omega(x) \leq \frac{\log |\Omega|}{p}, \tag{7}$$

where $|\cdot|$ represents the cardinality operator.

(iii) If the functions $f^j(\cdot)$, $j \in \Omega$, are continuously differentiable, then $\psi_{p\Omega}(\cdot)$ is continuously differentiable, with gradient

$$\nabla \psi_{p\Omega}(x) = \sum_{j \in \Omega} \mu_p^j(x) \nabla f^j(x), \tag{8}$$

where

$$\mu_p^j(x) \overset{\triangle}{=} \frac{\exp(pf^j(x))}{\sum\limits_{k \in \Omega} \exp(pf^k(x))} = \frac{\exp(p[f^j(x) - \psi_\Omega(x)])}{\sum\limits_{k \in \Omega} \exp(p[f^k(x) - \psi_\Omega(x)])} \in (0,1), \tag{9}$$

and $\sum_{j\in\Omega} \mu_p^j(x) = 1$ for all $x \in \mathbb{R}^d$.

(iv) If the functions $f^j(\cdot)$, $j \in \Omega$, are twice continuously differentiable, then $\psi_{p\Omega}(\cdot)$ is twice continuously differentiable, with Hessian

$$\begin{aligned} \nabla^2 \psi_{p\Omega}(x) &= \sum_{j \in \Omega} \mu_p^j(x) \nabla^2 f^j(x) + p \sum_{j \in \Omega} \mu_p^j(x) \nabla f^j(x) \nabla f^j(x)^T \\ &\quad -p \left[\sum_{j \in \Omega} \mu_p^j(x) \nabla f^j(x) \right] \left[\sum_{j \in \Omega} \mu_p^j(x) \nabla f^j(x) \right]^T \end{aligned} \tag{10}$$

for all $x \in \mathbb{R}^d$. □

We define a continuous, nonpositive optimality function $\theta_\Omega : \mathbb{R}^d \to \mathbb{R}$ for all $x \in \mathbb{R}^d$ by

$$\theta_\Omega(x) \overset{\triangle}{=} -\min_{\mu \in \Sigma_\Omega} \left\{ \sum_{j \in \Omega} \mu^j (\psi_\Omega(x) - f^j(x)) + \tfrac{1}{2} \left\| \sum_{j \in \Omega} \mu^j \nabla f^j(x) \right\|^2 \right\}, \tag{11}$$

where $\Sigma_\Omega \overset{\triangle}{=} \{\mu \in \mathbb{R}^{|\Omega|} \mid \mu^j \geq 0 \ \forall j \in \Omega, \sum_{j \in \Omega} \mu^j = 1\}$, which results in the following optimality condition for (P_Ω); see Theorems 2.1.1, 2.1.3, and 2.1.6 of [24].

Proposition 2.2. *Suppose that the functions $f^j(\cdot), j \in Q$, are continuously differentiable and that $\Omega \subseteq Q$. If $x^* \in \mathbb{R}^d$ is a local minimizer for (P_Ω), then $\theta_\Omega(x^*) = 0$.* □

The continuous, nonpositive optimality function $\theta_{p\Omega} : \mathbb{R}^d \to \mathbb{R}$ defined for all $x \in \mathbb{R}^d$ by $\theta_{p\Omega}(x) \overset{\triangle}{=} -\frac{1}{2}\|\nabla \psi_{p\Omega}(x)\|^2$ characterizes stationary points of $(P_{p\Omega})$.

3 Rate of Convergence and Run-Time Complexity

In this section, we examine the following basic smoothing algorithm, for which we develop a series of complexity and rate of convergence results. The algorithm applies the Armijo Gradient Method[1] to (P_p) for a fixed value of p.

Algorithm 3.1. Smoothing Armijo Gradient Algorithm

Data: Tolerance $t > 0, x_0 \in \mathbb{R}^d$.

Parameter: $\delta \in (0, 1)$.

Step 1. Set $p^* = (\log q)/((1-\delta)t)$.

Step 2. Generate a sequence $\{x_i\}_{i=0}^\infty$ by applying Armijo Gradient Method to (P_{p^*}). □

When they exist, we denote optimal solutions of (P) and (P_p) by x^* and x^*_p, respectively, and the corresponding optimal values by ψ^* and ψ^*_p. Algorithm 3.1 has the following property.

Proposition 3.1. *Suppose that $q \geq 2$ and Step 2 of Algorithm 3.1 has generated a point $x_i \in \mathbb{R}^d$ such that $\psi_{p^*}(x_i) - \psi^*_{p^*} \leq \delta t$. Then, $\psi(x_i) - \psi^* \leq t$.*

Proof. By the optimality of $\psi^*_{p^*}$ and (7), $\psi^*_{p^*} \leq \psi_{p^*}(x^*) \leq \psi^* + (\log q)/p^*$. Thus, $-\psi^* \leq -\psi^*_{p^*} + (\log q)/p^*$. Based on (7), $\psi(x_i) \leq \psi_{p^*}(x_i)$ and hence, $\psi(x_i) - \psi^* \leq \psi_{p^*}(x_i) - \psi^*_{p^*} + (\log q)/p^*$. Since $\psi_{p^*}(x_i) - \psi^*_{p^*} \leq \delta t$ and p^* is as in Step 1, the conclusion follows. □

For a fixed $p > 0$, the rate of convergence of the Armijo Gradient Method as applied to (P_p) is well known (see for example [24], p. 60). However, the value of the precision parameter p^* in Algorithm 3.1 is dictated by q and t (see Step 1), which complicates the analysis. For

[1]The Armijo Gradient Method uses the steepest descent search direction and the Armijo stepsize rule to solve an unconstrained problem; see for example Algorithm 1.3.3 of [24].

large values of q or small values of t, p^* is large and hence (P_{p^*}) may be ill-conditioned as observed empirically [7]. In this paper, we quantify the ill-conditioning of (P_p) as a function of p and obtain complexity and rate of convergence results for Algorithm 3.1.

3.1 Ill-Conditioning of Smoothed Problem

We examine the ill-conditioning of (P_p) under the following strong convexity assumption.

Assumption 3.1. *The functions $f^j(\cdot), j \in \mathbb{N}$, are*

(i) *twice continuously differentiable and*

(ii) *there exist an $m > 0$ such that*

$$m\|y\|^2 \leq \langle y, \nabla^2 f^j(x) y\rangle, \tag{12}$$

for all $x, y \in \mathbb{R}^d$, and $j \in \mathbb{N}$. □

Lemma 3.1. *Suppose that Assumption 3.1 holds. For any $x, y \in \mathbb{R}^d$, $q \in \mathbb{N}$, and $p > 0$,*

$$m\|y\|^2 \leq \left\langle y, \nabla^2 \psi_p(x) y\right\rangle, \tag{13}$$

with m as in Assumption 3.1.

Proof. From (10) and (12), we obtain that

$$\begin{aligned}
\left\langle y, \nabla^2 \psi_p(x) y\right\rangle &= \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla^2 f^j(x) y\right\rangle + p \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x) \nabla f^j(x)^T y\right\rangle \\
&- p \left\langle y, \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right]^T y \right\rangle \\
&= \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla^2 f^j(x) y\right\rangle + p \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x)\right\rangle^2 \\
&- p \left\langle y, \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \right\rangle^2 \\
&\geq m\|y\|^2 + p \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x)\right\rangle^2 - p \left\langle y, \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \right\rangle^2 .
\end{aligned}$$

Hence, we only need to show that the difference of the last two terms is nonnegative. Let $g : \mathbb{R}^d \to \mathbb{R}$ be the convex function defined as $g(z) = \langle y, z\rangle^2$ for $y, z \in \mathbb{R}^d$.

Hence, it follows from Jensen's inequality (see for example p. 6 of [25]) that

$$\sum_{j\in Q} \mu_p^j(x) g\left(\nabla f^j(x)\right) \geq g\left(\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)\right). \tag{14}$$

Since $p > 0$, the result follows. □

For any matrix $A \in \mathbb{R}^{m\times n}$, we adopt the matrix norm $\|A\| \overset{\triangle}{=} \max_{\|u\|=1} \|Au\|$, where $u \in \mathbb{R}^n$. Under Assumption 3.1(i), $|f^j(x)|$, $\|\nabla f^j(x)\|$, and $\|\nabla^2 f^j(x)\|$ are bounded on bounded subsets of $\mathbb{R}^d$ for given $j \in \mathbb{N}$. We also assume that the bounds are uniform across the family of functions as stated next, which holds for example under standard assumptions when $f^j(\cdot)$, $j \in \mathbb{N}$, arise from discretization of semi-infinite max functions.

Assumption 3.2. *For any bounded set $S \subset \mathbb{R}^d$, there exists a $K \in (0,\infty)$ such that $\max\{|f^j(x)|, \|\nabla f^j(x)\|, \|\nabla^2 f^j(x)\|\} \leq K$ for all $x \in S, j \in \mathbb{N}$.* □

Under this assumption, we obtain the following useful result.

Lemma 3.2. *Suppose that Assumptions 3.1(i) and 3.2 hold. Then, for every bounded set $S \subset \mathbb{R}^d$,*

$$\langle y, \nabla^2 \psi_p(x) y\rangle \leq pL\|y\|^2, \tag{15}$$

for all $x \in S, y \in \mathbb{R}^d$, $q \in \mathbb{N}$, and $p \geq 1$, where $L = K + 2K^2$, with K as in Assumption 3.2.

Proof. Recall that for matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times r}$, and vector $x \in \mathbb{R}^n$, we have that $\|Ax\| \leq \|A\|\|x\|$, $\|AB\| \leq \|A\|\|B\|$, and $\|xx^T\| = \|x\|^2$ (see for example p. 26 of [26]). We consider each of the three parts of $\nabla^2\psi_p(\cdot)$; see (10). Recall that $\sum_{j\in Q}\mu_p^j(x) = 1$ for all $x \in \mathbb{R}^d$, $q \in \mathbb{N}$, and $p > 0$. For any $x \in S$, $y \in \mathbb{R}^d$, and $q \in \mathbb{N}$, under Assumption 3.2, we obtain for the first part that

$$\begin{aligned}\left\langle y, \sum_{j\in Q}\mu_p^j(x)\nabla^2 f^j(x) y\right\rangle &\leq \|y\| \left\| \left(\sum_{j\in Q}\mu_p^j(x)\nabla^2 f^j(x)\right) y\right\| \\ &\leq \|y\|^2 \sum_{j\in Q}\mu_p^j(x)\left\|\nabla^2 f^j(x)\right\| \leq K\|y\|^2,\end{aligned} \tag{16}$$

where K is the constant in Assumption 3.2 corresponding to S. Next, for the second part of $\nabla^2\psi_p(\cdot)$,

$$\begin{aligned}
\left\langle y, \sum_{j\in Q} \mu_p^j(x) \nabla f^j(x) \nabla f^j(x)^T y \right\rangle &\leq \|y\|^2 \left\| \sum_{j\in Q} \mu_p^j(x) \nabla f^j(x) \nabla f^j(x)^T \right\| \\
&\leq \|y\|^2 \left(\sum_{j\in Q} \mu_p^j(x) \left\| \nabla f^j(x) \nabla f^j(x)^T \right\| \right) \\
&\leq K^2 \|y\|^2. \qquad (17)
\end{aligned}$$

For the third part, we obtain that

$$\begin{aligned}
&-\left\langle y, \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right]^T y \right\rangle \\
&\leq \|y\|^2 \left\| \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right]^T \right\| \leq K^2\|y\|^2. \qquad (18)
\end{aligned}$$

Hence, for all $x \in S$, $y \in \mathbb{R}^d$, $q \in \mathbb{N}$ and $p \geq 1$, $\langle y, \nabla^2\psi_p(x)y\rangle \leq (K + pK^2 + pK^2)\|y\|^2 \leq p(K + 2K^2)\|y\|^2$. □

Lemma 3.2 enables us to quantify the rate of convergence of the Armijo Gradient Method for (P_p), as a function of $p \geq 1$, which we consider next. Let $\mathbb{N}_0 \overset{\triangle}{=} \mathbb{N} \cup \{0\}$.

Proposition 3.2. *Suppose that Assumptions 3.1 and 3.2 hold. For any bounded set $S \subset \mathbb{R}^d$, there exists a $k \in (0,1)$ such that the rate of convergence for the Armijo Gradient Method to solve (P_p), initialized by $x_0 \in S$, is linear with coefficient $1 - k/p$ for any $p \geq 1$ and $q \in \mathbb{N}$. That is, for all sequence $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ generated by the Armijo Gradient Method when applied to (P_p), for any $p \geq 1$, $q \in \mathbb{N}$, and $x_0 \in S$, we have that*

$$\psi_p(x_{i+1}) - \psi_p^* \leq \left(1 - \frac{k}{p}\right)\left(\psi_p(x_i) - \psi_p^*\right) \quad \text{for all } i \in \mathbb{N}_0. \qquad (19)$$

Proof. It follows by Lemma 3.1 and Assumption 3.2, and the fact that $x_0 \in S$, that there exists a bounded set $S' \subset \mathbb{R}^d$ such that all sequences generated by Armijo Gradient Method on (P_p), initialized by $x_0 \in S$, are contained in S' for all $p \geq 1$, $q \in \mathbb{N}$, $x_0 \in S$. Let m be as in Assumption 3.1 and K be the constant in Assumption 3.2 corresponding to S'. In view of Lemmas 3.1 and 3.2,

$$m\|y\|^2 \leq \langle y, \nabla^2\psi_p(x)y\rangle \leq pL\|y\|^2, \qquad (20)$$

for all $x \in S'$, $y \in \mathbb{R}^d$, $q \in \mathbb{N}$, and $p \geq 1$, where $L = K + 2K^2$. Hence, we deduce from Theorem 1.3.7 of [24] that the rate of convergence for Armijo Gradient Method to solve (P_p)

is linear with coefficient $1 - 4m\beta\alpha(1-\alpha)/(pL) \in (0,1)$ for all $p \geq 1$, $q \in \mathbb{N}$, $x_0 \in S$, where $\alpha, \beta \in (0,1)$ are the Armijo line search parameters. Hence,

$$k = 4m\beta\alpha(1-\alpha)/L, \tag{21}$$

which is less than unity because $\alpha(1-\alpha) \in (0, 1/4]$ and $m \leq L$ in view of (20). $\square$

3.2 Complexity

The above results enable us to identify the run-time complexity of Algorithm 3.1 under the following assumption on the complexity of function and gradient evaluations. We let $t_0 \overset{\triangle}{=} \psi(x_0) - \psi^*$ for a given $x_0 \in \mathbb{R}^d$ and $q \in \mathbb{N}$.

Assumption 3.3. *There exist constants $a, b < \infty$ such that for any $d \in \mathbb{N}$, $j \in \mathbb{N}$, and $x \in \mathbb{R}^d$, the computational work to evaluate either $f^j(x)$ or $\nabla f^j(x)$ is no larger than ad^b.* $\square$

Theorem 3.1. *Suppose that Assumptions 3.1, 3.2, and 3.3 hold, and that Algorithm 3.1 terminates after n iterations with $\psi(x_n) - \psi^* \leq t$. Then, for any $d \in \mathbb{N}$ and bounded set $S \subset \mathbb{R}^d$, there exist constants $c, c', t' \in (0, \infty)$ such that the computational work until termination for Algorithm 3.1 is no larger than*

$$c\frac{q \log q \log \frac{c'}{\delta t}}{(1-\delta)t}, \tag{22}$$

for all $q \in \mathbb{N}, q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, t']$.

Proof. Let $q \geq 2$ and $t \in (0, \log q]$, which ensures that $p^* = (\log q)/[(1-\delta)t] > 1$. Thus, Proposition 3.2 applies and the number of iterations of the Armijo Gradient Method to generate $\{x_i\}_{i=0}^n$ such that $\psi_{p^*}(x_n) - \psi_{p^*}^* \leq \delta t$ is no larger than

$$\left\lceil \frac{\log \frac{\delta t}{t_0}}{\log(1 - \frac{k}{p^*})} \right\rceil, \tag{23}$$

where k is the constant in Proposition 3.2 corresponding to S and $\lceil \cdot \rceil$ denotes the ceiling operator. In view of Proposition 3.1, x_n also satisfies $\psi(x_n) - \psi^* \leq t$. Since the main computational work in each iteration for the Armijo Gradient Method is to determine $\nabla\psi_{p^*}(x_i)$, it follows by Assumption 3.3 that there exists $a, b < \infty$ such that the computational work in each iteration of the Armijo Gradient Method when applied to (P_{p^*}) is no larger than aqd^b.

Thus, the computational work in Algorithm 3.1 to termination at x_n is no larger than (23) multiplied by aqd^b. Let f^{1*} denote the minimum value of $f^1(\cdot)$, which is finite according to Assumption 3.1. Let K be the constant in Assumption 3.2 corresponding to S. We then find that $t_0 = \psi(x_0) - \psi^* \leq K - f^{1*} \overset{\triangle}{=} c'$, for any $x_0 \in S$ and $q \in \mathbb{N}$. It follows that the computational work in Algorithm 3.1 to termination at x_n is no larger than

$$aqd^b \left\lceil \frac{\log \frac{\delta t}{c'}}{\log(1 - \frac{k}{p^*})} \right\rceil \tag{24}$$

for any $q \in \mathbb{N}$, $q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, \log q]$. Since $\log x \leq x - 1$ for $x \in (0,1]$, it follows by the choice of p^* that the computational work in Algorithm 3.1 to termination at x_n is no larger than

$$aqd^b \left\lceil \frac{\log \frac{\delta t}{c'}}{\log\left(1 - \frac{k(1-\delta)t}{\log q}\right)} \right\rceil \leq aqd^b \left\lceil \frac{\log \frac{c'}{\delta t}}{\frac{k(1-\delta)t}{\log q}} \right\rceil, \tag{25}$$

for all $q \in \mathbb{N}, q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, \min\{\log q, c'\}]$.

There exists a $t' \in (0, \min\{\log q, c'\}]$ such that $\frac{\log q \log \frac{c'}{\delta t}}{k(1-\delta)t} \geq \frac{1}{2}$ for all $t \in (0, t']$, $q \in \mathbb{N}, q \geq 2$, and $\delta \in (0,1)$. This then implies that for all $q \in \mathbb{N}, q \geq 2$, $x_0 \in S$, $\delta \in (0,1)$, and $t \in (0, t']$,

$$aqd^b \left\lceil \frac{\log q \log \frac{c'}{\delta t}}{k(1-\delta)t} \right\rceil \leq 2aqd^b \left(\frac{\log q \log \frac{c'}{\delta t}}{k(1-\delta)t} \right) = \frac{2ad^b}{k} \left(\frac{q \log q \log \frac{c'}{\delta t}}{(1-\delta)t} \right). \tag{26}$$

Since k (see (21)) only depends on m from Assumption 3.1, K from Assumption 3.2, and user-defined parameters, the conclusion follows. □

We deduce from Theorem 3.1 and its proof that the number of iterations of Algorithm 3.1 required to achieve a solution with value within t of the optimal value of (P) is $O((1/t) \log 1/t)$ for fixed $q \geq 2$, $d \in \mathbb{N}$, and $\delta \in (0,1)$. This is worse than for example the Pshenichnyi-Pironneau-Polak (PPP) min-max algorithm (Algorithm 2.4.1 in [24]) and the modified conjugate gradient method in [17], pp. 282-283, which achieve $O(\log 1/t)$. The SQP algorithm in [6] may also require a low number of iterations as it converges superlinearly, but its complexity in t is unknown. The worse complexity for Algorithm 3.1 is caused by the fact that the Armijo Gradient Method exhibits slower rate of convergence as p increases (see Proposition 3.2) and a larger p is required in Algorithm 3.1 for a smaller t.

When we also include the work per iteration of Algorithm 3.1, we see from Theorem 3.1 that for fixed $t \in (0, t']$, $d \in \mathbb{N}$, and $\delta \in (0,1)$, the run-time complexity is $O(q \log q)$. For comparison, the run-time complexity of SQP and PPP algorithms to achieve a near-optimal solution of (P) is larger as we see next.

The main computational work in an iteration of a SQP algorithm involve solving a convex QP with $d+1$ variables and q inequality constraints [6]. Introducing slack variables to convert into standard form, this subproblem becomes a convex QP with $d+1+q$ variables and q equality constraints. Based on [27], the number of operations to solve the converted QP is $O((d+1+q)^3)$. Assuming that the number of iterations a SQP algorithm needs to achieve a near-optimal solution of (P) is $O(1)$, and again focusing on q, the run-time complexity of a SQP algorithm to achieve a near-optimal solution of (P) is no better than $O(q^3)$. The same result holds for the PPP algorithm. This complexity, when compared with $O(q \log q)$ of Algorithm 3.1, indicates that smoothing algorithms may be more efficient than SQP and PPP algorithms for (P) with large q. We carry out a comprehensive numerical comparison of smoothing algorithms with SQP and PPP algorithms in Section 5. We note that the modified conjugate gradient method in [17], pp. 282-283, may also have a low run-time complexity in q, but this depends on its implementation and the method is only applicable to convex problems.

3.3 Optimal Parameter Choice

We see from Theorem 3.1 that the computational work in Algorithm 3.1 depends on the algorithm parameter δ. In this subsection, we find an "optimal" choice of δ. A direct minimization of (22) with respect to δ appears difficult and thus, we carry out a rate analysis and determine an optimal δ in that context.

We first consider the situation as $t \downarrow 0$ and let $\delta_t \in (0,1)$ be a choice of δ in Algorithm 3.1 for a specific t. For fixed $d \in \mathbb{N}$, $q \in \mathbb{N}$, $q \geq 2$, $S \subset \mathbb{R}^d$, and $x_0 \in S$, let c and c' be as in Theorem 3.1 and let w_t denote (22) viewed as a function of $t > 0$, with δ replaced by δ_t, i.e.,

$$w_t \triangleq \tilde{c} \frac{\log \frac{c'}{\delta_t t}}{(1-\delta_t)t} \tag{27}$$

with $\tilde{c} = cq \log q$ for all $t > 0$. The next result shows that the choice of $\{\delta_t \in (0,1) \mid t > 0\}$ influences the rate with which $w_t \to \infty$, as $t \downarrow 0$. However, any constant δ_t for all $t > 0$ results in the slowest possible rate of increase in w_t, an asymptotic rate of $1/t$, as $t \downarrow 0$.

Theorem 3.2. *For any* $\{\delta_t \in (0,1) \mid t > 0\}$,

$$\limsup_{t\downarrow 0} \frac{\log w_t}{\log t} \leq -1. \tag{28}$$

If $\delta_t = a \in (0,1)$ *for all* $t > 0$*, then*

$$\lim_{t\downarrow 0} \frac{\log w_t}{\log t} = -1. \tag{29}$$

Proof. There exists a $t_1 \in (0,\infty)$ such that $\log \frac{c'}{\delta_t t} \geq 1$ for all $t \in (0, t_1]$, and any $\{\delta_t \in (0,1) \mid t > 0\}$. Hence, for any $t \in (0, \min\{1, t_1\})$ and $\delta_t \in (0,1)$,

$$\begin{aligned} \frac{\log w_t}{\log t} &= \frac{\log \tilde{c}}{\log t} + \frac{\log \log \frac{c'}{\delta_t t}}{\log t} - \frac{\log(1-\delta_t)}{\log t} - \frac{\log t}{\log t} \\ &\leq \frac{\log \tilde{c}}{\log t} - 1. \end{aligned} \tag{30}$$

and the first part follows. Taking limits in (30), with $\delta_t = a$, yields the second part. $\square$

We next consider the situation as $q \to \infty$ and, similar to above, let $\delta_q \in (0,1)$ be a choice of δ in Algorithm 3.1 for a specific $q \in \mathbb{N}$. For fixed $d \in \mathbb{N}$ and $S \subset \mathbb{R}^d$, let c and c' be as in Theorem 3.1. There exists a $t_1 \in (0,\infty)$ such that $\log(c/t) \geq 0$ and $\log(c'/t) \geq 1$ for all $t \in (0, t_1]$. For any given $q \in \mathbb{N}$, $q \geq 2$ and $t \in (0, t_1]$, let w_q denote (22) viewed as a function of q, with δ replaced by δ_q, i.e.,

$$w_q \triangleq \left(\frac{c}{t}\right) \frac{q \log q \log \frac{c'}{\delta_q t}}{(1-\delta_q)}. \tag{31}$$

The next result shows that the choice of $\{\delta_q\}_{q=2}^{\infty}$ influences the rate with which $w_q \to \infty$, as $q \to \infty$. However, for sufficiently small tolerance $t > 0$, as above, any constant choice of δ_q for all $q \in \mathbb{N}$ results in the slowest possible rate of increase in w_q, as $q \to \infty$. Hence, any constant $\delta \in (0,1)$ in Algorithm 3.1 is optimal in this sense and results in the asymptotic rate of q, as $q \to \infty$.

Theorem 3.3. *For any sequence of* $\{\delta_q\}_{q=3}^{\infty}$*, with* $\delta_q \in (0,1)$*, we have that*

$$\frac{\log w_q}{\log q} \geq 1, \tag{32}$$

for all $q \in \mathbb{N}$*,* $q \geq 3$*,* $t \in (0, t_1]$*. If* $\delta_q = a$*, where* $a \in (0,1)$ *is a constant, then*

$$\lim_{q\to\infty} \frac{\log w_q}{\log q} = 1. \tag{33}$$

Proof. For $q \geq 3$,

$$\begin{aligned} \frac{\log w_q}{\log q} &= \frac{\log \frac{c}{t}}{\log q} + \frac{\log q}{\log q} + \frac{\log\log q}{\log q} + \frac{\log\log \frac{c'}{\delta_q t}}{\log q} - \frac{\log(1-\delta_q)}{\log q} \\ &\geq \frac{\log \frac{c}{t}}{\log q} + 1 + \frac{\log\log \frac{c'}{\delta_q t}}{\log q}. \end{aligned} \tag{34}$$

Since w_q is defined only for $t \in (0, t_1]$, and $\log(c/t) \geq 0$ and $\log(c'/t) \geq 1$ for all $t \in (0, t_1]$, it follows that $(\log w_q)/\log q \geq 1$ for all $q \geq 3$, $t \in (0, t_1]$, and $\{\delta_q\}_{q=3}^{\infty}$. The proof for the second part follows from taking the limit in (34). □

3.4 Rate of Convergence

The previous subsection considers the effect of the algorithm parameter δ on the computational work required in Algorithm 3.1. This parameter defines the precision parameter through the relationship $p^* = (\log q)/((1-\delta)t)$; see Step 1 of Algorithm 3.1. In this subsection, we do not restrict Algorithm 3.1 to this class of choices for p^* and consider any positive value of the precision parameter. In particular, we examine the progress made by Algorithm 3.1 after n iterations for different choices of p^*. Since the choice may depend on n, we denote by p_n the precision parameter used in Algorithm 3.1 when terminated after n iterations. We examine the rate of decay of an error bound on $\psi(x_n) - \psi^*$, and also determine the "optimal choice" of p_n that produces the fastest rate of decay of the error bound as $n \to \infty$.

Suppose that Assumptions 3.1 and 3.2 hold. For a given bounded set $S \subset \mathbb{R}^d$, let k be as in Proposition 3.2 and let $\{x_i\}_{i=0}^{n}$, with $x_0 \in S$, be a sequence generated by Algorithm 3.1 using $p^* = p_n$ for some $p_n > 0$. Then, in view of (7) and Proposition 3.2,

$$\begin{aligned} \psi(x_n) - \psi^* &\leq \psi_{p_n}(x_n) - \psi_{p_n}^* + \frac{\log q}{p_n} \\ &\leq \left(1 - \frac{k}{p_n}\right)^n \left(\psi_{p_n}(x_0) - \psi_{p_n}^*\right) + \frac{\log q}{p_n} \\ &\leq \left(1 - \frac{k}{p_n}\right)^n \left(\psi(x_0) - \psi^*\right) + \frac{2\log q}{p_n}. \end{aligned} \tag{35}$$

We want to determine the "best" $\{p_n\}_{n=1}^{\infty}$ such that the error bound on $\psi(x_n) - \psi^*$ defined by the right-hand side of (35) decays as fast as possible as $n \to \infty$. We denote that error bound by e_n, i.e., for any $n \in \mathbb{N}$,

$$e_n \triangleq t_0 \left(1 - \frac{k}{p_n}\right)^n + \frac{2\log q}{p_n}. \tag{36}$$

We need the following trivial technical result.

Lemma 3.3. *For $x \in [0, 1/2]$, $-2x \leq \log(1 - x) \leq -x$.* □

We next obtain that e_n asymptotically decays with a rate no faster than $1/n$, as $n \to \infty$, regardless of the choice of p_n, and that rate is attained with a particular choice of p_n.

Theorem 3.4. *The following hold about e_n in (36):*

(i) For any $\{p_n\}_{n=1}^{\infty}$, with $p_n \geq 1$ for all $n \in \mathbb{N}$, $\liminf_{n\to\infty} \log e_n / \log n \geq -1$.

(ii) If $p_n = \zeta n / \log n$ for all $n \in \mathbb{N}$, with $\zeta \in (0, k]$, then $\lim_{n\to\infty} \log e_n / \log n = -1$.

(iii) If $p_n = n^{1-\nu} / \log n$ for all $n \in \mathbb{N}$, with $\nu \in (0, 1)$, then $\lim_{n\to\infty} \log e_n / \log n = -1 + \nu$.

Proof. See Appendix A. □

We see from Theorem 3.4 that the "best" choice of $p_n = \zeta n / \log n$, with $\zeta \in (0, k]$, and it results in an asymptotic rate of $1/n$. The constant k may be unknown as it depends on m of Assumption 3.1 and K of Assumption 3.2; see (21). Consequently, $p_n = \zeta n / \log n$ may be difficult to implement unless there are conservative estimates of m and K. Theorem 3.4 shows that the choice $p_n = n^{1-\nu} / \log n$ with a small $\nu \in (0, 1)$ is almost as good (it results in asymptotic rate $1/n^{1-\nu}$ instead of rate $1/n$) and is independent of k.

Roughly speaking, the error rate of no better than $1/n$ indicated by Theorem 3.4 translates to a rate of increase in the required number of iterations of at least $1/t$, where t is the stipulated error (tolerance). In view of Theorem 3.2, the rate $1/t$ is attained with the precision parameter choice in Step 1 of Algorithm 3.1. Hence, in some sense, the choice in Step 1 of Algorithm 3.1 for the precision parameter cannot be improved.

Theorems 3.2 and 3.4 indicate that Algorithm 3.1 may only converge sublinearly. In contrast, Theorem 3.1 shows that smoothing algorithms may still be capable of yielding competitive run times against other algorithms when q is large due to low computational effort per iteration. For smoothing algorithms to be competitive in empirical test, however, we need to go beyond the basic Algorithm 3.1 and develop more sophisticated, adaptive precision-adjustment schemes as discussed next.

4 Smoothing Algorithms and Adaptive Precision Adjustment

The previous section shows that the choice of precision parameter influences the rate of convergence as the degree of ill-conditioning in (P_p) depends on the precision parameter. In this section, we present two smoothing algorithms with novel precision-adjustment schemes for (P). In view of [7] and our preliminary numerical tests, we focus on adaptive precision-adjustment schemes as they appear superior to static and open-loop schemes in their ability to avoid ill-conditioning.

The first algorithm, Algorithm 4.1, is the same as Algorithm 3.2 in [13], but uses a much simpler scheme for precision adjustment. The second algorithm, Algorithm 4.2, adopts a novel line search rule that aims to ensure descent in $\psi(\cdot)$ and, if that is not possible, increases the precision parameter. Previous smoothing algorithms [7, 13] do not check for descent in $\psi(\cdot)$. The new algorithms implement active-set strategies adapted from [13].

We use the following notation. The ϵ-active set, $\epsilon > 0$, is denoted by

$$Q_\epsilon(x) \overset{\triangle}{=} \{j \in Q | \psi(x) - f^j(x) \leq \epsilon\}. \tag{37}$$

As in Algorithm 3.2 of [13], we compute a search direction using a $d \times d$ matrix $B_{p\Omega}(x)$. We consider two options. When

$$B_{p\Omega}(x) = I, \tag{38}$$

the $d \times d$ identity matrix, the search direction is equivalent to the steepest descent direction. When

$$B_{p\Omega}(x) = \eta_{p\Omega}(x) I + H_{p\Omega}(x), \tag{39}$$

the search direction is a Quasi-Newton direction, where

$$H_{p\Omega}(x) \overset{\triangle}{=} p \left(\sum_{j\in\Omega} \mu_p^j(x) \nabla f^j(x) \nabla f^j(x)^T - \left(\sum_{j\in\Omega} \mu_p^j(x) \nabla f^j(x) \right) \left(\sum_{j\in\Omega} \mu_p^j(x) \nabla f^j(x) \right)^T \right), \tag{40}$$

$$\eta_{p\Omega}(x) \overset{\triangle}{=} \max\{0, \varphi - e_{p\Omega}(x)\}, \tag{41}$$

$\varphi > 0$, and $e_{p\Omega}(x)$ is the smallest eigenvalue of $H_{p\Omega}(x)$. The quantity $\eta_{p\Omega}(x)$ ensures that $B_{p\Omega}(x)$ is positive definite. The Quasi-Newton direction given in (39)-(41) is adopted from [13]. As stated in [13], the justification for ignoring the first term of the Hessian function in (10) is the observation that when $p \to \infty$, the first term becomes negligible.

We next present the two algorithms and their proofs of convergence.

Algorithm 4.1.

Data: $x_0 \in \mathbb{R}^d$.

Parameters: $\alpha, \beta \in (0,1), p_0 \geq 1, \omega = (10 \log q)/p_0$, function $B_{p\Omega}(\cdot)$ as in (38) or (39), $\epsilon_0 > 0, \xi > 1, \varsigma > 1, \varphi \geq 1$.

Step 1. Set $i = 0, j = 0, \Omega_0 = Q_{\epsilon_0}(x_0)$.

Step 2. Compute the search direction $h_{p_i\Omega_i}(x_i)$ by solving the equation

$$B_{p_i\Omega_i}(x_i) h_{p_i\Omega_i}(x_i) = -\nabla \psi_{p_i\Omega_i}(x_i). \tag{42}$$

Step 3. Compute the stepsize $\lambda_i = \beta^{k_i}$, where k_i is the largest integer k such that

$$\psi_{p_i\Omega_i}(x_i + \beta^k h_{p_i\Omega_i}(x_i)) - \psi_{p_i\Omega_i}(x_i) \leq -\alpha\beta^k \|h_{p_i\Omega_i}(x_i)\|^2 \tag{43}$$

and

$$\psi_{p_i\Omega_i}(x_i + \beta^k h_{p_i\Omega_i}(x_i)) - \psi(x_i + \beta^k h_{p_i\Omega_i}(x_i)) \geq -\omega. \tag{44}$$

Step 4. Set

$$x_{i+1} = x_i + \beta^{k_i} h_{p_i\Omega_i}(x_i), \tag{45}$$

$$\Omega_{i+1} = \Omega_i \cup Q_{\epsilon_i}(x_{i+1}). \tag{46}$$

Step 5. Enter Subroutine 4.1, and go to Step 2 when exit Subroutine 4.1. □

Subroutine 4.1. Adaptive Precision-Parameter Adjustment using Optimality Function

If

$$\theta_{p_i\Omega_i}(x_{i+1}) \geq -\epsilon_i, \tag{47}$$

set $x_j^* = x_{i+1}$, set $p_{i+1} = \xi p_i$, set $\epsilon_{i+1} = \epsilon_i/\varsigma$, replace i by $i+1$, replace j by $j+1$, and exit Subroutine 4.1.

Else, set $p_{i+1} = p_i$, set $\epsilon_{i+1} = \epsilon_i$, replace i by $i+1$, and exit Subroutine 4.1. □

Steps 1 to 4 of Algorithm 4.1 are adopted from Algorithm 3.2 of [13]. We note the unusual choice of the right-hand side in (43), where $-\|h_{p_i\Omega_i}(x_i)\|^2$ is used instead of the conventional $\langle \nabla\psi_{p_i\Omega_i}(x_i), h_{p_i\Omega_i}(x_i)\rangle$. Test runs show that Algorithm 4.1 with $-\|h_{p_i\Omega_i}(x_i)\|^2$ is slightly more efficient than with the conventional $\langle \nabla\psi_{p_i\Omega_i}(x_i), h_{p_i\Omega_i}(x_i)\rangle$. To allow direct comparison with Algorithm 3.2 of [13], we use $-\|h_{p_i\Omega_i}(x_i)\|^2$ in Algorithm 4.1.

The test in (44) prevents the construction of a point x_{i+1} where $\psi(x_{i+1})$ is much greater than $\psi(x_i)$ during the early iterations when the set Ω_i is small; see [13].

The key difference between Algorithm 4.1 and Algorithm 3.2 of [13] is the simplified scheme to adjust p_i in Subroutine 4.1. This difference calls for a different proof of convergence as compared to [13], and will be based on *consistent approximation.* Let $\mathcal{P}$ denote an increasing sequence of positive real numbers that approach infinity. Modified to our context, we define consistent approximation as in [24], p. 399:

Definition 4.1. *For any $\Omega \subset \mathbb{N}$, we say that the pairs $((P_{p\Omega}), \theta_{p\Omega}(\cdot))$ in the sequence $\{((P_{p\Omega}), \theta_{p\Omega}(\cdot))\}_{p\in\mathcal{P}}$ are consistent approximations to $((P_\Omega), \theta_\Omega(\cdot))$, if (i) $(P_{p\Omega})$ epi-converges to (P_Ω), as $p \to^{\mathcal{P}} \infty$, and (ii) for any infinite sequence $\{x_p\}_{p\in\mathcal{P}_0} \subset \mathbb{R}^d$, $\mathcal{P}_0 \subset \mathcal{P}$, such that $x_p \to x^*$, $\limsup_{p\to\infty} \theta_{p\Omega}(x_p) \le \theta_\Omega(x^*)$.* □

Recall that epi-convergence of $(P_{p\Omega})$ to (P_Ω) refers to set convergence (in the sense of Painlevé-Kuratowski) of a sequence of epigraphs of $(P_{p\Omega})$ to the epigraph of (P_Ω); see section 3.3.1 of [24] and Sections 1B, 4B, and 7B of [28] for a detailed exposition of epi-convergence.

The following result is required in the proof of convergence of Algorithm 4.1.

Theorem 4.1. *Suppose that Assumption 3.1(i) holds. Then, for any $\Omega \subset \mathbb{N}$, the pairs $((P_{p\Omega}), \theta_{p\Omega}(\cdot))$ in the sequence $\{((P_{p\Omega}), \theta_{p\Omega}(\cdot))\}_{p\in\mathcal{P}}$ are consistent approximations to $((P_\Omega), \theta_\Omega(\cdot))$.*

Proof. We follow the proofs of Lemmas 4.3 and 4.4 in [29], but simplify the arguments as [29] deals with min-max-min problems. By Theorem 3.3.2 of [24], Proposition 2.1(ii), and the continuity of $\psi_\Omega(\cdot)$, it follows that $(P_{p\Omega})$ epi-converges to (P_Ω), as $p \to \infty$.

We next consider the optimality functions. Let $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ and $\{p_i\}_{i=0}^\infty, p_i > 0$ for all i, be arbitrary sequences and $x^* \in \mathbb{R}^d$ be such that $x_i \to x^*$ and $p_i \to \infty$, as $i \to \infty$. Since $\mu_p^j(x) \in (0, 1)$ for any $j \in \Omega$, $p > 0$, and $x \in \mathbb{R}^d$, $\{\mu_{p_i}(x_i)\}_{i=0}^\infty$ is a bounded sequence in $\mathbb{R}^{|\Omega|}$ with at least one convergent subsequence. For every such subsequence $K \subset \mathbb{N}_0$, there exists a $\mu_\infty \in \Sigma_\Omega$ such that $\mu_{p_i}(x_i) \to^K \mu_\infty$, as $i \to \infty$. Moreover, since $\mu_\infty \in \Sigma_\Omega$, $\sum_{j\in\Omega} \mu_\infty^j = 1$.

If $j \notin \hat{\Omega}(x^*)$, then there exist a $t > 0$ and $i_0 \in \mathbb{N}$ such that $f^j(x_i) - \psi_\Omega(x_i) \le -t$ for all $i \ge i_0$. Hence, from (9), $\mu_{p_i}^j(x_i) \to 0$, as $i \to \infty$, and therefore $\mu_\infty^j = 0$. By continuity of $\nabla f^j(\cdot)$, $j \in \Omega$,

$$\theta_{p_i\Omega}(x_i) \to^K -\frac{1}{2}\|\sum_{j\in\Omega} \mu_\infty^j \nabla f^j(x^*)\|^2 \triangleq \theta_{\infty\Omega}(x^*), \tag{48}$$

as $i \to \infty$. Since $\mu_\infty \in \Sigma_\Omega$ and $\mu_\infty^j = 0$ for all $j \notin \hat{\Omega}(x^*)$, we find in view of (11) that

$$\theta_{\infty\Omega}(x^*) \;=\; -\sum_{j\in\Omega} \mu_\infty^j(\psi_\Omega(x^*) - f^j(x^*)) - \frac{1}{2}\|\sum_{j\in\Omega}\mu_\infty^j \nabla f^j(x^*)\|^2 \leq \theta_\Omega(x^*). \qquad (49)$$

This completes the proof. □

The next result is identical to Lemma 3.1 in [13].

Lemma 4.1. *Suppose that $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ is a sequence constructed by Algorithm 4.1. Then, there exists an $i^* \in \mathbb{N}_0$ and a set $\Omega^* \subseteq Q$ such that working sets $\Omega_i = \Omega^*$ for all $i \geq i^*$.*

Proof. By construction, $\Omega_i \subseteq \Omega_{i+1}$ for all $i \in \mathbb{N}_0$. Since the set Q is finite, the lemma must be true. □

The following result is a proof of convergence of Algorithm 4.1.

Theorem 4.2. *Suppose that Assumption 3.1(i) holds. Then, any accumulation point $x^* \in \mathbb{R}^d$ of a sequence $\{x_j^*\}_{j=0}^\infty \subset \mathbb{R}^d$ constructed by Algorithm 4.1 satisfies the first-order optimality condition $\theta(x^*) = 0$.*

Proof. Let $\Omega^* \subseteq Q$ and $i^* \in \mathbb{N}_0$ be as in Lemma 4.1, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. As Algorithm 4.1 has the form of Master Algorithm Model 3.3.12 in [24] for all $i \geq i^*$, we conclude based on Theorem 3.3.13 in [24] that any accumulation point x^* of a sequence $\{x_j^*\}_{j=0}^\infty$ constructed by Algorithm 4.1 satisfies $\theta_{\Omega^*}(x^*) = 0$. The assumptions required to invoke Theorem 3.3.13 in [24] are (i) continuity of $\psi_{\Omega^*}(\cdot), \psi_{p\Omega^*}(\cdot), \theta_{\Omega^*}(\cdot)$, and $\theta_{p\Omega^*}(\cdot)$, $p > 0$, which follows by Assumption 3.1(i), Proposition 2.1(i), Theorem 2.1.6 of [24], and Proposition 2.1(iii); (ii) the pairs $((P_{p\Omega^*}), \theta_{p\Omega^*}(\cdot))$ in the sequence $\{((P_{p\Omega^*}), \theta_{p\Omega^*}(\cdot))\}_{p\in\mathcal{P}}$ are consistent approximations to $((P_{\Omega^*}), \theta_{\Omega^*}(\cdot))$, which follows by Theorem 4.1; and (iii) if Steps 1 to 4 of Algorithm 4.1 are applied repeatedly to $(P_{p\Omega^*})$ with a fixed $p > 0$, then every accumulation point $\hat{x}$ of a sequence $\{x_k\}_{k=0}^\infty$ constructed must be a stationary point of $(P_{p\Omega^*})$, i.e., $\theta_{p\Omega^*}(\hat{x}) = 0$, which follows by Theorem 3.2 in [13].

Since $\theta_{\Omega^*}(x^*) = 0$, from (11), there exists a $\mu \in \Sigma_{\Omega^*}$ such that

$$\sum_{j\in\Omega^*} \mu^j(\psi_{\Omega^*}(x^*) - f^j(x^*)) + \tfrac{1}{2}\left\|\sum_{j\in\Omega^*}\mu^j \nabla f^j(x^*)\right\|^2 = 0. \qquad (50)$$

Let $\pi \in \Sigma_Q$, $\pi^j = 0$ for $j \in Q - \Omega^*$, and $\pi^j = \mu^j$ for $j \in \Omega^*$. Thus, it follows from (11) that

$$\theta(x^*) \geq -\sum_{j\in Q} \pi^j(\psi(x^*) - f^j(x^*)) - \tfrac{1}{2}\left\|\sum_{j\in Q}\pi^j \nabla f^j(x^*)\right\|^2 = 0. \qquad (51)$$

Since $\theta(\cdot)$ is a nonpositive function, the result follows. $\square$

Algorithm 4.2.

Data: $x_0 \in \mathbb{R}^d$.

Parameters: $\alpha, \beta \in (0,1)$, function $B_{p\Omega}(\cdot)$ as in (38) or (39), $\epsilon > 0, \varphi \geq 1, p_0 \geq 1, \hat{p} \gg p_0, \kappa \gg 1, \xi > 1, \gamma > 0, \nu \in (0,1), \Delta p \geq 1$.

Step 0. Set $i = 0, \Omega_0 = Q_\epsilon(x_0), k_{-1} = 0$.

Step 1. Compute $B_{p_i\Omega_i}(x_i)$ and its largest eigenvalue $\sigma^{\max}_{p_i\Omega_i}(x_i)$. If

$$\sigma^{\max}_{p_i\Omega_i}(x_i) \geq \kappa, \tag{52}$$

compute the search direction

$$h_{p_i\Omega_i}(x_i) = -\nabla\psi_{p_i\Omega_i}(x_i). \tag{53}$$

Else, compute the search direction $h_{p_i\Omega_i}(x_i)$ by solving the equation

$$B_{p_i\Omega_i}(x_i)h_{p_i\Omega_i}(x_i) = -\nabla\psi_{p_i\Omega_i}(x_i). \tag{54}$$

Step 2a. Compute a tentative Armijo stepsize based on working set Ω_i, starting from the eventual stepsize of the previous iterate k_{i-1}, i.e., determine

$$\lambda_{p_i\Omega_i}(x_i) = \max_{l\in\{k_{i-1},k_{i-1}+1,\ldots\}}\{\beta^l|\psi_{p_i\Omega_i}(x_i+\beta^l h_{p_i\Omega_i}(x_i))-\psi_{p_i\Omega_i}(x_i) \leq \alpha\beta^l\langle\nabla\psi_{p_i\Omega_i}(x_i), h_{p_i\Omega_i}(x_i)\rangle\}. \tag{55}$$

Set

$$y_i = x_i + \beta^l h_{p_i\Omega_i}(x_i). \tag{56}$$

Step 2b. Forward track from y_i along direction $h_{p_i\Omega_i}(x_i)$ as long as $\psi(\cdot)$ continues to decrease using the following subroutine.

Substep 0. Set $l' = l$,

$$z_{il'} = x_i + \beta^{l'} h_{p_i\Omega_i}(x_i) \text{ and } z_{il'-1} = x_i + \beta^{l'-1}h_{p_i\Omega_i}(x_i). \tag{57}$$

Substep 1. If

$$\psi(z_{il'-1}) < \psi(z_{il'}), \tag{58}$$

replace l' by $l'-1$, set $z_{il'-1} = x_i + \beta^{l'-1}h_{p_i\Omega_i}(x_i)$, and repeat Substep 1.

Else, set $z_i = z_{il'}$.

Substep 2. If $p_i \leq \hat{p}$, go to Step 3. Else, go to Step 4.

Step 3. If

$$\psi(z_i) - \psi(x_i) \leq -\frac{\gamma}{{p_i}^{\nu}}, \tag{59}$$

set $x_{i+1} = z_i, p_{i+1} = p_i, k_i = l'$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1. Else, replace p_i by ξp_i, replace Ω_i by $\Omega_i \cup Q_\epsilon(z_i)$, and go to Step 1.

Step 4. If (59) holds, set $x_{i+1} = z_i, k_i = l'$, set $p_{i+1} = p_i + \Delta p$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1.

Else, set $x_{i+1} = y_i, k_i = l$, set $p_{i+1} = p_i + \Delta p$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1. □

As is standard in stabilized Newton methods (see for example Section 1.4.4 of [24]), Algorithm 4.2 switches to the steepest descent direction if $B_{p\Omega}(\cdot)$ is given by (39) and the largest eigenvalue of $B_{p\Omega}(\cdot)$ is large; see Step 1. Compared to Algorithm 3.2 in [13], which increases p when $\|\nabla\psi_{p_i\Omega_i}(x_i)\|$ is small, Algorithm 4.2 increases the precision parameter only when it does not produce sufficient descent in $\psi(\cdot)$, as verified by the test (59) in Steps 3 and 4 of Algorithm 4.2. A small precision parameter may produce an ascent direction in $\psi(\cdot)$ due to the poor accuracy of $\psi_{p_i\Omega_i}(\cdot)$. Thus, insufficient descent is a signal that the precision parameter may be too small. All existing smoothing algorithms only ensure that $\psi_{p_i\Omega_i}(\cdot)$ decreases at each iteration, but do not ensure descent in $\psi(\cdot)$. Another change compared to [7, 13] relates to the line search. All smoothing algorithms are susceptible to ill-conditioning and small stepsizes. To counteract this difficulty, Algorithm 4.2 moves forward along the search direction starting from the Armijo step, and stops when the next step is not a descent step in $\psi(\cdot)$; see Step 2b.

Algorithm 4.2 has two rules for increasing p_i. In the early stages of the calculations, i.e., when $p_i \leq \hat{p}$, if sufficient descent in $\psi(\cdot)$ is achieved when moving from x_i to z_i ((59) satisfied), then Algorithm 4.2 sets the next iterate x_{i+1} to z_i, retain the current value of the precision parameter as progress is made towards the optimal solution of (P). However, if (59) fails, then there is insufficient descent and the precision parameter or the working set needs to be modified to generate a better search direction in the next iteration. In late stages of the calculations, i.e., $p_i > \hat{p}$, Algorithm 4.2 accepts every new point generated, even those with insufficient descent, and increases the precision parameter with a constant value.

The next lemma is similar to Lemma 4.1.

Lemma 4.2. *Suppose that $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ is a sequence constructed by Algorithm 4.2. Then, there exists an $i^* \in \mathbb{N}_0$ and a set $\Omega^* \subseteq Q$ such that working sets $\Omega_i = \Omega^*$ and $\psi_{\Omega^*}(x_i) = \psi(x_i)$ for all $i \geq i^*$.*

Proof. The first part of the proof follows exactly from the proof for Lemma 4.1. Next, since $\widehat{Q}(x_i) \subseteq \Omega_i$ for all i; see Steps 3 and 4 of Algorithm 4.2, $\psi_{\Omega^*}(x_i) = \psi(x_i)$ for all $i \geq i^*$. □

Lemma 4.3. *Suppose that Assumption 3.1(i) holds, and that the sequences $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ and $\{p_i\}_{i=0}^{\infty} \subset \mathbb{R}$ are generated by Algorithm 4.2. Then, the following properties hold: (i) the sequence $\{p_i\}_{i=0}^{\infty}$ is monotonically increasing; (ii) if the sequence $\{x_i\}_{i=0}^{\infty}$ has an accumulation point, then $p_i \to \infty$ as $i \to \infty$, and $\sum_{i=0}^{\infty} 1/p_i = +\infty$.*

Proof. We follow the framework of the proof for Lemma 3.1 of [7]. (i) The precision parameter is adjusted in Steps 3 and 4 of Algorithm 4.2. In Step 3, if (59) is satisfied, then $p_{i+1} = p_i$; if (59) fails, p_i is replaced by $\xi p_i > p_i$. In Step 4, $p_{i+1} = p_i + \Delta p \geq p_i + 1 > p_i$.

(ii) Suppose that Algorithm 4.2 generates the sequence $\{x_i\}_{i=0}^{\infty}$ with accumulation point $x^* \in \mathbb{R}^d$, but $\{p_i\}_{i=0}^{\infty}$ is bounded from above. The existence of an upper bound on p_i implies that $p_i \leq \hat{p}$ for all $i \in \mathbb{N}_0$, because if not, Algorithm 4.2 will enter Step 4 the first time at some iteration $i' \in \mathbb{N}_0$, and re-enter Step 4 for all $i > i'$, and $p_i \to \infty$ as $i \to \infty$. Thus, the existence of an upper bound on p_i implies that Algorithm 4.2 must never enter Step 4.

The existence of an upper bound on p_i also implies that there exist an iteration $i^* \in \mathbb{N}_0$ such that (59) is satisfied for all $i > i^*$, because if not, p_i will be replaced by ξp_i repeatedly, and $p_i \to \infty$ as $i \to \infty$. This means that $\psi(x_{i+1}) - \psi(x_i) \leq -\gamma/{p_i}^{\nu}$ for all $i > i^*$. Since $p_i \leq \hat{p}$ for all $i \in \mathbb{N}_0$, $\psi(x_i) \to -\infty$ as $i \to \infty$. However, by continuity of $\psi(\cdot)$, and x^* being an accumulation point, $\psi(x_i) \to^K \psi(x^*)$, where $K \subset \mathbb{N}_0$ is some infinite subset. This is a contradiction, so $p_i \to \infty$.

Next, we prove that $\sum_{i=0}^{\infty} 1/p_i = +\infty$. Since $p_i \to \infty$, there exist an iteration $i^* \in \mathbb{N}_0$ such that $p_i > \hat{p}$ for all $i \geq i^*$. This means that the precision parameter is adjusted by the rule $p_{i+1} = p_i + \Delta p$ for all $i \geq i^*$. The proof is complete by the fact that $\sum_{i=1}^{\infty} 1/i = \infty$. □

Lemma 4.4. *Suppose that Assumption 3.1(i) holds. Then, for every bounded set $S \subset \mathbb{R}^d$ and parameters $\alpha, \beta \in (0,1)$, there exist a $K < \infty$ such that, for all $p \geq 1$, $\Omega \subseteq Q$, and*

$x \in S$,

$$\psi_{p\Omega}(x + \lambda_{p\Omega}(x)h_{p\Omega}(x)) - \psi_{p\Omega}(x) \leq \frac{-\alpha K\|\nabla\psi_{p\Omega}(x)\|^2}{p}, \tag{60}$$

where $\lambda_{p\Omega}(x)$ is the stepsize defined by (55), with p_i replaced by p, Ω_i replaced by Ω, and x_i replaced by x.

Proof. If $h_{p\Omega}(x)$ is given by (54) with $B_{p\Omega}(x)$ as in (38), then the result follows by the same arguments as in the proof for Lemma 3.2 of [7]. If $h_{p\Omega}(x)$ is given by (54) with $B_{p\Omega}(x)$ as in (39), then the result follows by similar arguments as in the proof for Lemma 3.4 of [7], but the argument deviates to account for the fact that the lower bound on the eigenvalues of $B_{p\Omega}(x)$ takes on the specific value of 1 in Algorithm 4.2. $\square$

Lemma 4.5. *Suppose that Assumption 3.1(i) holds and that $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ is a bounded sequence generated by Algorithm 4.2. Let $\Omega^* \subseteq Q$ and $i^* \in \mathbb{N}_0$ be as in Lemma 4.2, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. Then, there exist an accumulation point $x^* \in \mathbb{R}^d$ of the sequence $\{x_i\}_{i=0}^\infty$ such that $\theta_{\Omega^*}(x^*) = 0$.*

Proof. Suppose that $\{x_i\}_{i=0}^\infty$ is a bounded sequence generated by Algorithm 4.2. Suppose that there exist an $\rho > 0$ such that

$$\liminf_{i\to\infty} \|\nabla\psi_{p_i\Omega^*}(x_i)\| \geq \rho. \tag{61}$$

Since $\{x_i\}_{i=0}^\infty$ is a bounded sequence, it has at least one accumulation point. Hence, by Lemma 4.3, $p_i \to \infty$, as $i \to \infty$. Consider two cases, $x_{i+1} = y_i$ or $x_{i+1} = z_i$ in Algorithm 4.2. If $x_{i+1} = y_i$, by Lemma 4.4, there exist an $M < \infty$ such that

$$\psi_{p_i\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq -\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \tag{62}$$

for $i \geq i^*$. Hence,

$$\begin{aligned}\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) &= \psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_{i+1}) + \psi_{p_i\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \\ &\leq -\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i},\end{aligned} \tag{63}$$

for $i \geq i^*$, where we have used the fact from Proposition 2.1 that

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) \leq \psi_{p_i\Omega^*}(x_{i+1}), \tag{64}$$

for $i \geq i^*$, because $p_{i+1} \geq p_i$ from Lemma 4.3.

Next, if $x_{i+1} = z_i$, then (59) is satisfied. It follows from (7) and Lemma 4.2 that,

$$\begin{aligned}\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) &\leq \psi_{\Omega^*}(x_{i+1}) + \frac{\log|\Omega^*|}{p_{i+1}} - \psi_{\Omega^*}(x_i) \\ &= \psi(x_{i+1}) + \frac{\log|\Omega^*|}{p_{i+1}} - \psi(x_i) \\ &\leq -\frac{\gamma}{{p_i}^{\nu}} + \frac{\log|\Omega^*|}{p_i} \\ &= \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^{\nu}}. \end{aligned} \tag{65}$$

From (63) and (65), for all $i \geq i^*$,

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq \max\left\{-\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^{\nu}}\right\} \tag{66}$$

By Proposition 2.1, $\|\nabla\psi_{p_i\Omega^*}(x_i)\|$ is bounded because $\{x_i\}_{i=0}^{\infty}$ is bounded. Since $\nu \in (0,1)$, there exist an $i^{**} \in \mathbb{N}_0$, where $i^{**} \geq i^*$, such that

$$-\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i} \geq \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^{\nu}}, \tag{67}$$

for all $i \geq i^{**}$. Therefore, from (66),

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq -\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \tag{68}$$

for all $i \geq i^{**}$. Since by Lemma 4.3, $\sum_{i=0}^{\infty} 1/p_i = +\infty$, it follows from (63) and (68) that

$$\psi_{p_i\Omega^*}(x_i) \to -\infty, \text{ as } i \to \infty. \tag{69}$$

Let x^* be an accumulation point of $\{x_i\}_{i=0}^{\infty}$. That is, there exist an infinite subset $K \subseteq \mathbb{N}_0$ such that $x_i \to^K x^*$. Based on (7), Lemma 4.3, and continuity of $\psi_{\Omega^*}(\cdot)$, it follows that $\psi_{p_i\Omega^*}(x_i) \to^K \psi_{\Omega^*}(x^*)$, as $i \to \infty$, which contradicts (69). Hence, $\liminf_{i\to\infty} \|\nabla\psi_{p_i\Omega^*}(x_i)\| = 0$. Consequently, there exists an infinite subset $K^* \subseteq \mathbb{N}_0$ and an $x^* \in \mathbb{R}^d$ such that $x_i \to x^*$ and $\theta_{p_i\Omega^*}(x_i) \to^{K^*} 0$, as $i \to \infty$, which implies that $\limsup_{i\to\infty} \theta_{p_i\Omega^*}(x_i) \geq 0$. From Definition 4.1, Theorem 4.1, and the fact that $\theta_{\Omega^*}(\cdot)$ is a nonpositive function, $\theta_{\Omega^*}(x^*) = 0$. □

Theorem 4.3. *Suppose that Assumption 3.1(i) holds. (i) If Algorithm 4.2 constructs a bounded sequence $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$, then there exists an accumulation point $x^* \in \mathbb{R}^d$ of the sequence $\{x_i\}_{i=0}^{\infty}$ that satisfies $\theta(x^*) = 0$. (ii) If Algorithm 4.2 constructs a finite sequence*

$\{x_i\}_{i=0}^{i^*} \subset \mathbb{R}^d$, where $i^* < \infty$, then Step 2b constructs an unbounded infinite sequence $\{z_{i^*l'}\}_{l'=l}^{-\infty}$ with

$$\psi(z_{i^*l'-1}) < \psi(z_{i^*l'}), \tag{70}$$

for all $l' \in \{l, l-1, l-2, ...\}$, where l is the tentative Armijo stepsize computed in Step 2a.

Proof. First, we consider (i). Let the set $\Omega^* \subseteq Q$ be as in Lemma 4.2, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. Based on Lemma 4.5, there exist an accumulation point of the sequence $\{x_i\}_{i=0}^{\infty}$, $x^* \in \mathbb{R}^d$ such that $\theta_{\Omega^*}(x^*) = 0$. The conclusion then follows by similar arguments as in Theorem 4.2.

We next consider (ii). Algorithm 4.2 constructs a finite sequence only if it jams in Step 2b. Then, Substep 1 constructs an infinite sequence $\{z_{i^*l'}\}_{l'=l}^{-\infty}$ satisfying (70) for all $l' \in \{l, l-1, l-2, ...\}$. The infinite sequence is unbounded since $h_{p_i\Omega_i}(x_i) \neq 0$ as (70) cannot hold otherwise, and $\beta \in (0, 1)$. $\square$

Next, we consider the run-time complexity in q for a fixed $d \in \mathbb{N}$ of Algorithms 4.1 and 4.2 to achieve a near-optimal solution of (P). Suppose that all functions $f^j(\cdot)$ are active, i.e., $\Omega_i = Q$, near an optimal solution. If $B_{p\Omega}(\cdot)$ is given by (38), then the main computational work in each iteration of Algorithms 4.1 and 4.2 is the calculation of $\nabla\psi_p(\cdot)$, which takes $O(q)$ operations under Assumption 3.3; see the proof of Theorem 3.1. If $B_{p\Omega}(\cdot)$ is given by (39), then the main computational work is the calculation of (39) and $h_{p\Omega}(x)$. Under Assumption 3.3, it takes $O(q)$ operations to compute $\mu_p^j(x)$, $j \in Q$, $O(q)$ to compute $\nabla f^j(x)$, $j \in Q$, $O(q)$ to sum $\sum_{j\in\Omega} \mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T$, $O(q)$ to sum $\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)$, and the other operations take $O(1)$. In all, the number of operations to obtain $B_{p\Omega}(x)$ is $O(q)$. A direct method for solving a linear system of equations to compute $h_{p\Omega}(x)$ depends on d, but is constant in q. Hence, if $B_{p\Omega}(\cdot)$ is given by (39), then the computational work in each iteration of Algorithms 4.1 and 4.2 is $O(q)$. It is unclear how many iterations Algorithms 4.1 and 4.2 would need to achieve a near-optimal solution as a function of q. However, since they may utilize Quasi-Newton search directions and adaptive precision adjustment, there is reason to believe that the number of iterations will be no larger than that of Algorithm 3.1, which uses the steepest descent direction and a fixed precision parameter. Thus, suppose that for some tolerance $t > 0$, the number of iterations of Algorithms 4.1 and 4.2 to generate

$\{x_i\}_{i=0}^n$, with the last iterate satisfying $\psi(x_n) - \psi^* \leq t$, is no larger than $O(\log q)$, as is the case for Algorithm 3.1. Then, the run-time complexity of Algorithms 4.1 and 4.2 to generate x_n is no larger than $O(q \log q)$, which is the same as for Algorithm 3.1.

5 Numerical Results

We present an empirical comparison of Algorithms 4.1 and 4.2 with algorithms from the literature over a set of problem instances from [6, 7] as well as randomly generated instances; see Appendix B and Table 1. This study appears to be the first systematic comparison of smoothing and SQP algorithms for large-scale problems, with up to two orders of magnitude larger q than previously reported.

Specifically, we examine (i) Algorithm 2.1 of [6], an SQP algorithm with two QPs that we refer to as SQP-2QP, (ii) Algorithm A in [9], a one-QP SQP algorithm that we refer to as SQP-1QP, (iii) Algorithm 3.2 in [13], a smoothing Quasi-Newton algorithm referred to as SMQN, (iv) Pshenichnyi-Pironneau-Polak min-max algorithm (Algorithm 2.4.1 in [24]), referred to as PPP, (v) an active-set version of PPP as stated in Algorithm 2.4.34 in [24]; see also [30], which we refer to as ϵ-PPP, and (vi) Algorithms 4.1 and 4.2 of the present paper. We refer to Appendix C for details about algorithm parameters. With the exception of PPP and SQP-1QP, the above algorithms incorporate active-set strategies and, hence, appear especially promising for solving problem instances with large q. We implement and run all algorithms in MATLAB version 7.7.0 (R2008b) (see [31]) on a 3.73 GHz PC using Windows XP SP3, with 3 GB of RAM. All QPs are solved using TOMLAB CPLEX version 7.0 (R7.0.0) (see [32]) with the Primal Simplex option, which preliminary studies indicate result in the smallest QP run time. We also examined the LSSOL QP solver (see [33]), but its run times appear inferior to that of CPLEX for large-scale QPs arising in the present context.

Algorithm 2.1 of [6] is implemented in the solver CFSQP [34] and we have verified that our MATLAB implementation of that algorithm produces comparable results in terms of number of iterations and run time as CFSQP. We do not directly compare with CFSQP as we find it more valuable to compare different algorithms using the same implementation environment (MATLAB) and the same QP solver (CPLEX).

We carry out a comprehensive study to identify an ϵ (see (37)) in the algorithms' active-set strategies that minimizes the run time for the various algorithms over a wide range of ϵ (1,000 to 10^{-20}). We find that SQP-2QP is insensitive to the selection of ϵ, primarily because the algorithm includes additional steps to aggressively trim the working set. ϵ-PPP is highly sensitive to ϵ with variability within a factor of 200 in run times. SMQN, Algorithm 4.1, and Algorithm 4.2 accumulate functions in the working set and therefore are also sensitive to ϵ. The run times of SMQN, Algorithm 4.1, and Algorithm 4.2 tend to vary within a factor of ten. The below results are obtained using the apparent, best choice of ϵ for each algorithm.

For Algorithm 4.2, we mainly use the Quasi-Newton direction with $B_{p\Omega}(x)$ as defined in (39), because preliminary test runs show that generally, the alternate steepest descent direction with $B_{p\Omega}(x)$ as defined in (38) produces slower run times. We examine all problem instances from [6, 7] except two that cannot be easily extended to large q. As the problem instances with large dimensionality in [6, 7] do not allow us to adjust the number of functions, we create two additional sets of problem instances; see Appendix B for details. We report run times to achieve a solution x that satisfies

$$\psi(x) - \psi^{\text{target}} \leq t, \tag{71}$$

where ψ^{target} is a target value (see Table 1) equal to the optimal value (if known) or a slightly adjusted value from the optimal values reported in [6, 7] for smaller q. We use $t = 10^{-5}$. Although this termination criteria is not possible for real-world problems, we find that it is the most useful criterion in this study.

Table 2 summarizes the run times (in seconds) of the various algorithms, with columns 2 and 3 giving the number of variables d and functions q, respectively. Run times in boldface indicate that the particular algorithm has the shortest run time for the specific problem instance. The numerical results in Table 2 indicate that in most problem instances, the run times are shortest for SQP-2QP or Algorithm 4.2. Table 2 indicates that SQP-2QP is significantly more efficient than SQP-1QP for problem instances ProbA-ProbG. This is due to the efficiency of the active-set strategy in SQP-2QP, which is absent in SQP-1QP. However, for ProbJ-ProbM, SQP-1QP is comparable to SQP-2QP. This is because at the optimal solution of ProbJ-ProbM, all the functions are active. This causes the active-set strategy in SQP-2QP to lose its effectiveness as the optimal solution is approached.

Table 2 indicates also that Algorithm 4.1 is significantly more efficient than SMQN for most problem instances. As the only difference between the two algorithms lie in their precision-parameter adjustment scheme, this highlights the sensitivity in the performance of smoothing algorithms to the control of their precision parameters. Table 2 also shows that Algorithm 4.2 is more efficient than Algorithm 4.1 and SMQN for most problem instances.

Table 2 indicates that SQP-2QP is generally more efficient than Algorithm 4.2 for problem instances with small dimensionality, $d \leq 4$ (specifically ProbA-ProbG), and vice versa. This is consistent with the common observation that SQP-type algorithms may be inefficient for instances of large dimensionality; see for example [6].

Table 2 shows that some algorithms return locally optimal solutions for some problem instances (labeled “local” in Table 2). In view of these results, there is an indication that smoothing algorithms (SMQN, Algorithms 4.1 and 4.2) tend to find global minima more frequently than PPP and SQP algorithms.

Table 3 presents similar results as in Table 2, but for larger q. We do not present results for PPP and SQP-1QP as the required QPs exceed the memory limit. The comprehensive sensitivity studies for ϵ show significant improvement for Algorithm 4.2 for ProbJ-ProbM if a large ϵ is used. Hence, we include the results for Algorithm 4.2 with $\epsilon = 1000$ in Table 3. This ϵ-value means that there is effectively no active-set strategy. Sensitivity tests conducted for the other algorithms with a larger ϵ show no improvement in their run times.

The observations from Table 3 are similar to those for Table 2. Table 3 indicates that Algorithm 4.2 with $\epsilon = 1000$ is efficient for ProbJ-ProbM, which has large d and a significant number of functions active at the optimal solution. For completeness, the run times for Algorithm 4.2 with $\epsilon = 1000$ for ProbJ-ProbM in Table 2 are 2.8, 14.3, 0.36 and 3.0 seconds respectively, while the run times for the other problem instances are slower than Algorithm 4.2 with $\epsilon = 10^{-20}$.

The results in Tables 2 and 3 indicate that among the algorithms considered, SQP-2QP and Algorithm 4.2 are the most efficient algorithms for minimax problems with a large number of functions. The run times for ProbJ-ProbM indicate that SQP-2QP is less efficient for problem instances with a significant number of the functions that is nearly active at the solution, as the active-set strategy loses its effectiveness.

The problem instances from the literature examined in Tables 2 and 3 include either cases with few functions ϵ-active at an optimal solution (ProbA-ProbI) or cases with all functions ϵ-active (ProbJ-ProbM). We also examine randomly-generated problem instances with an intermediate number of functions ϵ-active at the optimal solution; see ProbN in Table 1. The optimal values are unknown in this case but the target values given in Table 1 appear to be close to the global minima.

Table 4 presents the run times for Algorithm 4.2 and SQP-2QP on ProbN. As the problem instances are relatively well-conditioned, Algorithm 4.2 with $B_{p\Omega}(\cdot)$ given by (38), i.e., a steepest descent (SD) direction, may perform well and is included in the table. The parameter ϵ for Algorithm 4.2 is set to 1000 for this set of problem instances, as preliminary test runs show that it is consistently better than other choices. Table 4 indicates that SQP-2QP is less efficient than Algorithm 4.2 for problem instances with large d, and where there is a significant number of functions ϵ-active at the optimal solution. The last row in Table 4 shows that for problem instances with $d \geq 10,000$, the storage of the $d \times d$ $H_{p\Omega}(\cdot)$ matrix for both SQP-2QP and Algorithm 4.2, with $B_{p\Omega}(\cdot)$ given by (39), causes both algorithms to terminate due to memory limitations. Thus, Algorithm 4.2, with $B_{p\Omega}(\cdot)$ given by (38), which do not have any matrix to store, may be a reasonable alternative when d is large.

6 Conclusions

This paper focuses on finite minimax problems with many functions and presents complexity and rate of convergence analysis of smoothing algorithms for such problems. We find that smoothing algorithms may only have sublinear rate of convergence, but their run-time complexity in the number of functions is competitive with other algorithms due to small computational work per iteration. We present two smoothing algorithms with novel precision-adjustment schemes and carry out a comprehensive numerical comparison with other algorithms from the literature. We find that the proposed algorithms are competitive, and especially efficient for problem instances with a significant number of functions nearly active at stationary points.

References

[1] E. Polak. On the Mathematical Foundations of Nondifferentiable Optimization in Engineering Design. *SIAM Review*, 29:21–89, 1987.

[2] E. Polak, S. Salcudean, and D. Q. Mayne. Adaptive Control of ARMA Plants using Worst Case Design by Semi-Infinite Optimization. *IEEE Transactions on Automatic Control*, 32:388–397, 1987.

[3] X. Cai, K. Teo, X. Yang, and X. Zhou. Portfolio Optimization Under a Minimax Rule. *Management Science*, 46(7):957–972, 2000.

[4] V. F. Demyanov and V. N. Malozemov. *Introduction to Minimax.* Wiley, New York, New York, 1974.

[5] E. R. Panier and A. L. Tits. A Globally Convergent Algorithm with Adaptively Refined Discretization for Semi-Infinite Optimization Problems arising in Engineering Design. *IEEE Transactions on Automatic Control*, 34(8):903–908, 1989.

[6] J. L. Zhou and A. L. Tits. An SQP Algorithm for Finely Discretized Continuous Minimax Problems and other Minimax Problems with Many Objective Functions. *SIAM J. Optimization*, 6(2):461–487, 1996.

[7] E. Polak, J. O. Royset, and R. S. Womersley. Algorithms with Adaptive Smoothing for Finite Minimax Problems. *J. Optimization Theory and Applications*, 119(3):459–484, 2003.

[8] E. Obasanjo, G. Tzallas-Regas, and B. Rustem. An Interior-Point Algorithm for Nonlinear Minimax Problems. *J. Optimization Theory and Applications*, 144:291–318, 2010.

[9] Z. Zhu, X. Cai, and J. Jian. An Improved SQP Algorithm for Solving Minimax Problems. *Applied Mathematics Letters*, 22(4):464–469, 2009.

[10] J. F. Sturm and S. Zhang. A Dual and Interior-Point Approach to Solve Convex Min-Max Problems. In D. Z. Du and P. M. Pardalos, editors, *Minimax and Applications*, pages 69–78. Kluwer Academic Publishers, 1995.

[11] L. Luksan, C. Matonoha, and J. Vlcek. Primal Interior-Point Method for Large Sparse Minimax Optimization. Technical Report 941, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 2005.

[12] F. Ye and H. Liu and S. Zhou and S. Liu. A Smoothing Trust-Region Newton-CG Method for Minimax Problem. *Applied Mathematics and Computation*, 199(2):581–589, 2008.

[13] E. Polak, R. S. Womersley, and H. X. Yin. An Algorithm Based on Active Sets and Smoothing for Discretized Semi-Infinite Minimax Problems. *J. Optimization Theory and Applications*, 138:311–328, 2008.

[14] X. Li. An Entropy-Based Aggregate Method for Minimax Optimization. *Engineering Optimization*, 18:277–285, 1992.

[15] S. Xu. Smoothing Method for Minimax Problems. *Computational Optimization and Applications*, 20:267–279, 2001.

[16] B. W. Kort and D. P. Bertsekas. A New Penalty Function Algorithm for Constrained Minimization. *Proceedings 1972 IEEE Conf. Decision and Control*, (82):343–362, 1972.

[17] A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* John Wiley, New York, 1983.

[18] Z. Drezner. On the Complexity of the Exchange Algorithm for Minimax Optimization Problems. *Mathematical Programming*, 38(2):219–222, 1987.

[19] E. J. Wiest and E. Polak. On the Rate of Convergence of Two Minimax Algorithms. *Journal of Optimization Theory and Applications*, 71(1):1–30, 1991.

[20] Yu. Nesterov. Complexity Estimates of Some Cutting Plane Methods based on the Analytic Barrier. *Mathematical Programming*, 69(1):149–176, 1995.

[21] K. A. Ariyawansa and P. L. Jiang. On Complexity of the Translational-Cut Algorithm for Convex Minimax Problems. *Journal of Optimization Theory and Applications*, 107(2):223–243, 2000.

[22] Yu. Nesterov and J. Ph. Vial. Augmented Self-Concordant Barriers and Nonlinear Optimization Problems with Finite Complexity. *Mathematical Programming*, 99(1):149–174, 2004.

[23] Yu. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization)*. Kluwer Academic Publishers, Norwell, MA, 2004.

[24] E. Polak. *Optimization. Algorithms and Consistent Approximations.* Springer, New York, New York, 1997.

[25] H. Urruty and J. Baptiste. *Convex Analysis and Minimization Algorithms 1. Fundamentals.* Springer, Berlin, 1996.

[26] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization.* Addison-Wesley, Redwood, CA, 1991.

[27] R. D. C. Monteiro and I. Adler. Interior Path Following Primal-Dual Algorithms. Part II: Convex Quadratic Programming. *Mathematical Programming*, 44(1):43–66, 1989.

[28] R. T. Rockafellar and R. J. B. Wets. *Variational Analysis.* Springer, Heidelberg, 1998.

[29] E. Polak. Smoothing Techniques for the Solution of Finite and Semi-Infinite Min-Max-Min Problems. In G. D. Pillo and A. Murli, editors, *High Performance Algorithms and Software for Nonlinear Optimization.* Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

[30] E. Polak. On the Convergence of the Pshenichnyi-Pironneau-Polak Minimax Algorithm with an Active Set Strategy. *Journal of Optimization Theory and Applications*, 138(2):305–309, 2008.

[31] Mathworks Inc. *MATLAB 7 Getting Started Guide.* Natick, MA, 2009.

[32] Tomlab Optimization Inc. *User's Guide for TOMLAB/CPLEX v12.1.* Pullman, WA, 2009.

[33] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for LSSOL (Version 1.0): a Fortran Package for Constrained Linear Least-Squares and Convex Quadratic Programming.* Stanford, CA, 1986.

[34] C. Lawrence, J. L. Zhou, and A. L. Tits. User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. Technical report, 1997.

[35] R. A. Brualdi. *Introductory Combinatorics.* Prentice-Hall, Upper Saddle River, NJ, 2004.

[36] Z. Zhu and K. Zhang. A Superlinearly Convergent Sequential Quadratic Programming Algorithm for Minimax Problems. *Chinese Journal of Numerical Mathematics and Applications*, 27(4):15–32, 2005.

Appendix A. Proof of Theorem 3.4

Proof. For any $n \in \mathbb{N}$, we see from (36) that

$$\begin{aligned}\log e_n &= \log\left(\exp\left[\log t_0 + n\log\left(1-\frac{k}{p_n}\right)\right] + \frac{2\log q}{p_n}\right)\\ &\geq \log\left(\max\left\{\exp\left[\log t_0 + n\log\left(1-\frac{k}{p_n}\right)\right], \frac{2\log q}{p_n}\right\}\right)\\ &= \max\left\{\log\left(\exp\left[\log t_0 + n\log\left(1-\frac{k}{p_n}\right)\right]\right), \log\frac{2\log q}{p_n}\right\}.\end{aligned}$$

Hence, for any $n \in \mathbb{N}$, $n > 1$,

$$\frac{\log e_n}{\log n} \geq \max\left\{\frac{\log t_0}{\log n} + \frac{n\log\left(1-\frac{k}{p_n}\right)}{\log n}, -\frac{\log p_n}{\log n} + \frac{\log 2}{\log n} + \frac{\log\log q}{\log n}\right\}. \tag{72}$$

Let $\epsilon > 0$ be arbitrary. Then, there exists a $n_0 \in \mathbb{N}$ such that $(\log\log q)/\log n \geq -\epsilon$ for all $n \geq n_0$. If $(\log p_n)/\log n \leq 1$ and $n \geq \max\{2, n_0\}$, then

$$\frac{\log e_n}{\log n} \geq -\frac{\log p_n}{\log n} + \frac{\log 2}{\log n} + \frac{\log\log q}{\log n} \geq -\frac{\log p_n}{\log n} - \epsilon \geq -1-\epsilon. \tag{73}$$

Alternatively, suppose that $(\log p_n)/\log n > 1$. Hence, $n/p_n < 1$, and if $n \geq 2k$, then $k/p_n \in (0, 1/2]$. Based on Lemma 3.3 and (72),

$$\frac{\log e_n}{\log n} \geq \frac{\log t_0}{\log n} + \frac{n\log\left(1-\frac{k}{p_n}\right)}{\log n} \geq \frac{\log t_0}{\log n} + \frac{n\left(-\frac{2k}{p_n}\right)}{\log n} \geq \frac{\log t_0}{\log n} - \frac{2k}{\log n} \tag{74}$$

for all $n \geq 2k$ such that $(\log p_n)/\log n > 1$. Thus, there exists $n_1 \geq \max\{n_0, 2k\}$ such that

$$\frac{\log t_0}{\log n} - \frac{2k}{\log n} \geq -1 - \epsilon \tag{75}$$

for all $n \geq n_1$. Hence, for all $n \geq n_1$, $(\log e_n)/\log n \geq -1-\epsilon$. Since ϵ is arbitrary, the first part follows. Next, we prove the second part of the theorem. From (36), with $p_n = \zeta n/\log n$, where $\zeta \in (0, k]$,

$$\log e_n = \log\left(\exp\left[\log t_0 + n\log\left(1 - \frac{k\log n}{\zeta n}\right)\right] + \frac{2\log q \log n}{\zeta n}\right). \tag{76}$$

There exists a $n_2 \in \mathbb{N}$ such that $(k\log n)/\zeta n \in [0, 1/2]$ for all $n \geq n_2$. Thus, by Lemma 3.3,

$$\begin{aligned} &\log\left(\exp\left[\log t_0 + n\left(-\frac{2k\log n}{\zeta n}\right)\right] + \frac{2\log q\log n}{\zeta n}\right) \leq \log e_n \\ \leq\ &\log\left(\exp\left[\log t_0 + n\left(-\frac{k\log n}{\zeta n}\right)\right] + \frac{2\log q\log n}{\zeta n}\right) \end{aligned} \tag{77}$$

for all $n \geq n_2$. We first consider the lower bound in (77),

$$\begin{aligned} &\log\left(\exp\left[\log t_0 + n\left(-\frac{2k\log n}{\zeta n}\right)\right] + \frac{2\log q\log n}{\zeta n}\right) \\ =\ &\log\left(\frac{2\log q\log n}{\zeta n}\left[\frac{\exp\left(\log t_0 + \log n^{-2k/\zeta}\right)}{\frac{2\log q\log n}{\zeta n}} + 1\right]\right) \\ =\ &\log\left(\frac{2\log q\log n}{\zeta n}\right) + \log\left(\frac{t_0\zeta n^{1-\frac{2k}{\zeta}}}{2\log q\log n} + 1\right). \end{aligned} \tag{78}$$

Since $\zeta \in (0, k]$ and by continuity of the $\log(\cdot)$ function,

$$\lim_{n\to\infty}\log\left(\frac{t_0\zeta n^{1-\frac{2k}{\zeta}}}{2\log q\log n} + 1\right) = 0. \tag{79}$$

Continuing from (78), and using (79), we obtain that

$$\begin{aligned} &\lim_{n\to\infty}\frac{\log\left(\frac{2\log q\log n}{\zeta n}\right) + \log\left(\frac{t_0\zeta n^{1-\frac{2k}{\zeta}}}{2\log q\log n} + 1\right)}{\log n} \\ =\ &\lim_{n\to\infty}\frac{\log 2 + \log\log q + \log\log n - \log\zeta - \log n}{\log n} = -1. \end{aligned} \tag{80}$$

Similar arguments yield that the upper bound in (77) also tends to -1, as $n \to \infty$. Hence, the second conclusion follows. The third part of the theorem follows by similar arguments.$\square$

Appendix B. Problem Instances

Table 1 describes the problem instances used. Most columns are self-explanatory. Columns 2 and 3 give the number of variables d and functions q, respectively. The target values (column 7) are equal to the optimal values (if known) or a slightly adjusted value from the optimal values reported in [6, 7] for smaller q. The same target values are used for ProbA-ProbM in Tables 2 and 3.

In this appendix, we denote components of $x \in \mathbb{R}^d$ by subscripts, i.e., $x = (x_1, x_2, ..., x_d) \in \mathbb{R}^d$. When the problem is given in semi-infinite form, as in (82a) - (82i), the set Y is discretized into q equally spaced points if

$$\psi(x) = \max_{y \in Y} \phi(x, y), \tag{81a}$$

and $q/2$ equally spaced points if

$$\psi(x) = \max_{y \in Y} |\phi(x, y)|. \tag{81b}$$

ProbA is defined by (81a) and (82a), and ProbB-ProbI by (81b) and (82b)-(82i), respectively.

$$\phi(x, y) = (2y^2 - 1)x + y(1 - y)(1 - x), \quad Y = [0, 1] \tag{82a}$$

$$\phi(x, y) = (1 - y^2) - (0.5x^2 - 2yx), \quad Y = [-1, 1], \tag{82b}$$

$$\phi(x, y) = y^2 - (yx_1 + x_2 \exp(y)), \quad Y = [0, 2], \tag{82c}$$

$$\phi(x, y) = \frac{1}{1 + y} - x_1 \exp(yx_2), \quad Y = [-0.5, 0.5], \tag{82d}$$

$$\phi(x, y) = \sin y - (y^2 x_3 + yx_2 + x_1), \quad Y = [0, 1], \tag{82e}$$

$$\phi(x, y) = \exp(y) - \frac{x_1 + yx_2}{1 + yx_3}, \quad Y = [0, 1], \tag{82f}$$

$$\phi(x, y) = \sqrt{y} - [x_4 - (y^2 x_1 + yx_2 + x_3)^2], \quad Y = [0.25, 1], \tag{82g}$$

$$\phi(x, y) = \frac{1}{1 + y} - [x_1 \exp(yx_3) + x_2 \exp(yx_4)], \quad Y = [-0.5, 0.5], \tag{82h}$$

$$\phi(x, y) = \frac{1}{1 + y} - [x_1 \exp(yx_4) + x_2 \exp(yx_5) + x_3 \exp(yx_6)], \quad Y = [-0.5, 0.5], \tag{82i}$$

ProbJ-ProbM are defined by $\psi(x) = \max_{j\in Q} f^j(x)$, with $f^j(x)$ as in (82j)-(82m), respectively.

$$f^j(x) = x_j^2, \quad j = \{1, ..., q\}, \tag{82j}$$

$$f^j(x) = x_{(j-1)2+1}^2 + x_{2j}^2, \quad j = \{1, ..., q\}, \tag{82k}$$

$$f^j(x) = x_{(j-1)4+1}^2 + x_{(j-1)4+2}^2 + x_{(j-1)4+3}^2 + x_{4j}^2, \quad j = \{1, ..., q\}, \tag{82l}$$

$$f^j(x) = x_{k_j}^2 + x_{l_j}^2, \quad j = \left\{1, 2, 3, ..., \binom{d}{2}\right\}, \tag{82m}$$

where (k_j, l_j) are all 2-combinations (see Section 3.3 of [35]) of $\{1, 2, 3, ..., d\}$, and

$$f^j(x) = a_j x_i^2 + b_j x_i + c_j, j = \{1, ..., q\}, \tag{82n}$$

where $i = \left\lceil \frac{j}{q/d} \right\rceil$, and a_j, b_j, c_j are randomly generated from a uniform distribution on $[0.5, 1]$.

Appendix C. Algorithm Details and Parameters

PPP. Pshenichnyi-Pironneau-Polak min-max algorithm (Algorithm 2.4.1 in [24]) use $\alpha = 0.5, \beta = 0.8$, and $\delta = 1$. We use the same Armijo parameters α and β for all algorithms.

ϵ-PPP. ϵ-Active PPP algorithm (Algorithm 2.4.34 in [24]; see also [30]) use the same parameters as above. We implement the most recent version [30].

SQP-2QP. Sequential Quadratic Programming with two QPs in each iteration (Algorithm 2.1 of [6]) use parameters recommended in [6] and monotone line search. (We examined the use of nonmonotone line search in CFSQP, but find it inferior to monotone line search on the set of problem instances.)

SQP-1QP. Sequential Quadratic Programming with one QP in each iteration (Algorithm A in [9]) use mid-point values stated in Algorithm A, $\alpha = 0.25$ (not the Armijo parameter), $\tau = 2.5$, and $H_0 = I$. The same settings for α and H_0 are used by a co-author in [36].

SMQN. Smoothing Quasi-Newton algorithm (Algorithm 3.2 in [13]) use $p_0 = 1$, $B(\cdot) = I$, and Parameter Adjustment subroutine version “Case (A)” of [7].

Algorithm 4.1. This algorithm uses the same parameters as SMQN, except for in the Adaptive Penalty Parameter Adjustment subroutine, where it uses $\xi = 2, \varsigma = 2$.

Algorithm 4.2. This algorithm use parameters $t = 10^{-5}, p_0 = 1, \hat{p} = (\log q/t) \cdot 10^{10}, \kappa = 10^{30}, \xi = 2, \gamma = t \cdot 10^{-10}, \nu = 0.5, \Delta p = 10$.

Table 1: Problem instances. An asterisk * indicates that the problem instance are created by the authors. Refer to Appendix B for details.

Instance	d	q	$\psi(x)$	Convexity	Initial point	Target value	Ref.
ProbA	1	varies	(81a), (82a)	Convex	5	0.1783942	[7]
ProbB	1	varies	(81b), (82b)	Non-convex	1	1.0000100	[6]
ProbC	2	varies	(81b), (82c)	Convex	$(1, 1)$	0.5382431	[6]
ProbD	2	varies	(81b), (82d)	Non-convex	$(1, -1)$	0.0871534	[6]
ProbE	3	varies	(81b), (82e)	Convex	$(1, 1, 1)$	0.0045048	[7]
ProbF	3	varies	(81b), (82f)	Non-convex	$(1, 1, 1)$	0.0042946	[6]
ProbG	4	varies	(81b), (82g)	Non-convex	$(1, 1, 1, 1)$	0.0026500	[7]
ProbH	4	varies	(81b), (82h)	Non-convex	$(1, 1, -3, -1)$	0.0020688	[6]
ProbI	6	varies	(81b), (82i)	Non-convex	$(1, 1, 1, -7, -3, -1)$	0.0006242	[6]
ProbJ	q	varies	(2), (82j)	Convex	$(\frac{2}{q}, \frac{4}{q}, \frac{6}{q}, ..., 1, -1-\frac{2}{q}, ..., -2)$	0	[7]
ProbK	$2q$	varies	(2), (82k)	Convex	$(\frac{1}{q}, \frac{2}{q}, \frac{3}{q}, ..., 1, -1-\frac{1}{q}, ..., -2)$	0	[7]
ProbL	$4q$	varies	(2), (82l)	Convex	$(\frac{1}{2q}, \frac{2}{2q}, \frac{3}{2q}, ..., 1, -1-\frac{1}{2q}, ..., -2)$	0	[7]
ProbM	varies	$\binom{d}{2}$	(2), (82m)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0	*
ProbN(i)	10	10,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9278640	*
ProbN(ii)	100	10,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9313887	*
ProbN(iii)	$1,000$	10,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9288089	*
ProbN(iv)	10	100,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9307828	*
ProbN(v)	100	100,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9340950	*
ProbN(vi)	$1,000$	100,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9366594	*
ProbN(vii)	$1,000$	1,000,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9358776	*
ProbN(viii)	$1,000$	10,000,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9369501	*
ProbN(ix)	$10,000$	100,000	(2), (82n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1-\frac{2}{d}, ..., -2)$	0.9335266	*

Table 2: Run times (in seconds) for various algorithms. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (71), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (71) after 6 hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 10^{-3}$ at termination.

Instance	d	q	PPP	ϵ-PPP	SQP-2QP	SQP-1QP	SMQN	Algo 4.1	Algo 4.2
				$(\epsilon = 10^{-3})$	$(\epsilon = 1)$		$(\epsilon = 10^{-20})$	$(\epsilon = 10^{-20})$	$(\epsilon = 10^{-20})$
ProbA	1	100,000	17.3	2.5	0.45	13.7	0.64	0.41	**0.31**
ProbB	1	100,000	2.5	0.69	**0.06**	131.1	0.31	0.70	0.45
ProbC	2	100,000	15.3	5.0	**0.67**	11.9	12.6	1.9	1.3
ProbD	2	100,000	5.0	7.4	**0.21**	9.9	7.2	1.7	1.5
ProbE	3	100,000	19.5	14.5	**0.59**	18.3	8.1	2.2	2.0
ProbF	3	100,000	28.7	18.8	2.3	24.4	18.2	2.9	**2.1**
ProbG	4	100,000	local	local	**2.4**	79.1	25.3	28.7	20.1
ProbH	4	100,000	211.8	968.4	local	128.9	36.2	31.5	**23.5**
ProbI	6	100,000	37.7	local	local	**31.7**	425.2	512.9	local
ProbJ	1,000	1,000	*	*	1458	1161	**	2212	**465.6**
ProbK	2,000	1,000	*	*	**	8404	**	10620	**2265**
ProbL	400	100	**3.6**	79.7	15.1	14.5	112.0	17.5	4.6
ProbM	100	4950	7.0	160.3	2.7	3.6	4.5	3.6	**2.3**

Table 3: Similar results as in Table 2, but with larger q. The word “local” means that the algorithm converges to a locally optimal solution that does not satisfy (71), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (71) after 6 hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 0.01$ at termination.

Instance	d	q	ϵ-PPP	SQP-2QP	SMQN	Algo 4.1	Algo 4.2	Algo 4.2
			$(\epsilon = 10^{-3})$	$(\epsilon = 1)$	$(\epsilon = 10^{-20})$	$(\epsilon = 10^{-20})$	$(\epsilon = 1000)$	$(\epsilon = 10^{-20})$
ProbA	1	1,000,000	22.5	4.6	4.4	**2.7**	8.6	3.1
ProbB	1	1,000,000	6.1	**0.61**	2.7	4.8	2.5	3.0
ProbC	2	1,000,000	59.4	**7.2**	131.0	15.0	61.9	13.1
ProbD	2	1,000,000	79.3	**2.2**	75.3	12.3	47.5	13.4
ProbE	3	1,000,000	245.0	**5.5**	93.0	12.1	74.5	17.5
ProbF	3	1,000,000	332.9	22.9	185.9	21.7	74.1	**18.1**
ProbG	4	1,000,000	local	**27.2**	257.8	220.1	12227	169.5
ProbH	4	1,000,000	12322	local	362.8	240.4	4157	**238.4**
ProbI	6	1,000,000	local	local	4262	4016	**3717**	local
ProbJ	4,000	4,000	**	**	**	**	**92.8**	**
ProbK	4,000	2,000	**	**	**	**	**91.8**	**
ProbL	4,000	1,000	*	**	**	**	**106.6**	13273
ProbM	200	19900	*	24.7	66.1	917.3	8.6	**2.5**

Table 4: Run times (in seconds) of algorithms on problem instance ProbN. "SD" and "QN" indicate that Algorithm 4.2 uses $B_{p\Omega}(\cdot)$ given by (38) and (39), respectively. The word "mem" indicates that the algorithm terminates due to insufficient memory.

d	q	SQP-2QP ($\epsilon = 1$)	Algo 4.2 SD ($\epsilon = 1000$)	Algo 4.2 QN ($\epsilon = 1000$)
10	10,000	**0.42**	0.64	0.62
100	10,000	0.82	**0.48**	0.54
1,000	10,000	124.9	**0.38**	4.8
10	100,000	4.1	**3.8**	4.2
100	100,000	11.5	**3.8**	4.1
1,000	100,000	mem	**4.3**	9.7
1,000	1,000,000	mem	**37.2**	42.5
1,000	10,000,000	mem	**421.8**	492.5
10,000	100,000	mem	**6.3**	mem