Computational Optimization and Applications (2019) 73:159–199
https://doi.org/10.1007/s10589-019-00071-3

Fast bundle-level methods for unconstrained and ball-constrained convex optimization

Yunmei Chen[1] · Guanghui Lan[2] · Yuyuan Ouyang[3] · Wei Zhang[1]

Received: 3 June 2017 / Published online: 7 February 2019

Abstract

In this paper, we study a special class of first-order methods, namely bundle-level (BL) type methods, which can utilize historical first-order information through cutting plane models to accelerate the solutions in practice. Recently, it has been shown in Lan (149(1–2):1–45, 2015) that an accelerated prox-level (APL) method and its variant, the uniform smoothing level (USL) method, have optimal iteration complexity for solving black-box and structured convex programming (CP) problems without requiring input of any smoothness information. However, these algorithms require the assumption on the boundedness of the feasible set and their efficiency relies on the solutions of two involved subproblems. Some other variants of BL methods which could handle unbounded feasible set have no iteration complexity provided. In this work we develop the fast APL (FAPL) method and fast USL (FUSL) method that can significantly improve the practical performance of the APL and USL methods in terms of both computational time and solution quality. Both FAPL and FUSL enjoy the same optimal iteration complexity as APL and USL, while the number of subproblems in each iteration is reduced from two to one, and an exact method is presented to solve the only subproblem in these algorithms. Furthermore, we introduce a generic algorithmic framework to solve unconstrained CP problems through solutions to a series of ball-constrained CP problems that also exhibits optimal iteration complexity. Our numerical results on solving some large-scale least squares problems and total variation based image reconstructions have shown advantages of these new BL type methods over APL, USL, and some other first-order methods.

Keywords Convex programming · First-order · Optimal method · Bundle-level method · Total variation · Image reconstruction

December, 2014. This research was partially supported by NSF Grants CMMI-1254446, DMS-1319050, DMS-1719932, and ONR Grant N00014-13-1-0036.

✉ Guanghui Lan
george.lan@isye.gatech.edu

Extended author information available on the last page of the article

Mathematics Subject Classification 90C25 · 90C06 · 90C22 · 49M37

1 Introduction

Many data analysis problems are often modeled as the following broad class of convex programming (CP) problems:

$$f^* := \min_{x \in X} f(x), \tag{1.1}$$

where $X \subseteq \mathbb{R}^n$ is a closed convex set, and $f : X \to \mathbb{R}$ is a convex function. We denote by X^* the solution set of the above problem. Throughout this paper, we assume that the solution set X^* is nonempty. One example of problem (1.1) is the classic ridge regression model (namely, Tikhonov regularization) in statistical learning estimates parameters β by

$$\min_{\beta \in \mathbb{R}^n} \|y - A\beta\|^2 \text{ subject to } \|\beta\| \leq \lambda, \tag{1.2}$$

where $\|\cdot\|$ denotes the Euclidean norm, y describes the observed outcome, A are the predictors in the observed data, and λ is a regularization parameter. The above model can be viewed as a special case of (1.1) with $x = \beta$, $f(x) = \|y - Ax\|^2$, and $X = \{x \in \mathbb{R}^n : \|x\| \leq \lambda\}$. Another important example is the classical two-dimensional total variation (TV) based image reconstruction problem [2,3] given by:

$$\min_{u \in \mathbb{R}^n} \frac{1}{2}\|Au - b\|^2 + \lambda \|u\|_{TV}, \tag{1.3}$$

where A is the measurement matrix, u is the n-vector form of a two-dimensional image to the constructed, b represents the observed data, $\|\cdot\|_{TV}$ is the discrete TV semi-norm, and λ is the regularization parameter. Problem (1.3) can also be casted as (1.1) by setting $x = u$, $f(x) = \|Ax - b\|^2/2 + \lambda\|x\|_{TV}$, and $X = \mathbb{R}^n$. It is worth noting that while problem (1.2) has an Euclidean ball constraint, problem (1.3) is an unconstrained CP problem. Moreover, the objective function in (1.2) is smooth, while the objective function in (1.3) is defined as the summation of a smooth term $\|Ax - b\|^2/2$ and a nonsmooth term $\lambda\|x\|_{TV}$.

Due to the high dimensionality of x for many applications in data analysis and imaging, much recent research effort has been directed to the development of efficient first-order methods for solving (1.1). First-order methods use gradients (or subgradients) of f exclusively and hence possess significantly reduced iteration cost than second-order methods. The efficiency of these algorithms are often measured by their iteration complexity in terms of the number of (sub)gradient evaluations required to find an approximate solution of (1.1). In view of the classic complexity theory [4], for any first-order methods the number of (sub)gradient evaluations required to find an ϵ-solution of (1.1) (i.e., a point $x_\epsilon \in \mathbb{R}^n$ satisfying $f(x_\epsilon) - f^* \leq \epsilon$) cannot be smaller than $\mathcal{O}(1/\epsilon^2)$ if f is nonsmooth. This can be achieved, for example, by traditional subgradient methods. For a smooth f, the optimal iteration complexity is $\mathcal{O}(1/\sqrt{\epsilon})$, which can be achieved, for example, by Nesterov's accelerated gradient

(AG) algorithms [5–7]. Recently, by adapting Nesterov's AG schemes and smoothing technique [5], several popular classes of first-order methods have been developed to improve their iteration complexity bounds. For instance, the accelerated primal dual (APD) algorithm [8] and accelerated hybrid proximal extragradient algorithm [9], which exhibit the optimal iterative complexity for solving a broad class of saddle point problems, and several variants of alternating direction method of multipliers (ADMM)[10–15], have improved the iteration complexity regarding to the smooth component.

In this paper, we focus on a different class of first-order methods, i.e., bundle-level (BL) type methods, which can utilize historical first-order information through cutting plane models to accelerate the numerical performance of the gradient descent type methods as mentioned above. We first give a review on several different types of BL methods.

1.1 Cutting plane, bundle and bundle-level methods

The bundle-level method originated from the well-known Kelley's cutting-plane method in 1960 [16]. Consider the convex programming problem

$$f_X^* := \min_{x \in X} f(x), \tag{1.4}$$

where X is a compact convex set and f is a closed convex function. The fundamental idea of the cutting plane method is to generate a sequence of piecewise linear functions to approximate f on X. In particular, given $x_1, x_2, \ldots, x_k \in X$, this method approximates f by

$$m_k(x) := \max\{h(x_i, x), 1 \le i \le k\}, \tag{1.5}$$

and computes the iterate x_{k+1} by

$$x_{k+1} \in \mathrm{Argmin}_{x \in X} m_k(x), \tag{1.6}$$

where

$$h(z, x) := f(z) + \langle f'(z), x - z \rangle, \tag{1.7}$$

and $f'(x) \in \partial f(x)$, where $\partial f(x) := \{\xi \in \mathbb{R}^n | f(y) \ge f(x) + \langle \xi, y - x \rangle, \ \forall y \in \mathbb{R}^n\}$ denotes the subdifferential of f at x. Clearly, the functions m_i, $i = 1, 2, \ldots$, satisfy $m_i(x) \le m_{i+1}(x) \le f(x)$ for any $x \in X$, and are identical to f at those search points x_i, $i = 1, \ldots, k$. However, the inaccuracy and instability of the piecewise linear approximation m_k over the whole feasible set X may affect the selection of new iterates, and the above scheme converges slowly both theoretically and practically [4,7]. Some important improvements of Kelley's method have been made under the name of bundle methods (see, e.g., [17–21]). In particular, by incorporating the level sets into Kelley's method, Lemaréchal, Nemirovskii and Nesterov [20] proposed in 1995 the classic bundle-level (BL) method by performing a series of projections over the approximate level sets.

Given $x_1, x_2, \ldots, x_k$, the classic BL iteration consists of the following three steps:

(a) Set $\overline{f}_k := \min\{f(x_i), 1 \le i \le k\}$ and compute a lower bound on f_X^* by $\underline{f}_k = \min_{x \in X} m_k(x)$.
(b) Set the level $l_k = \beta \underline{f}_k + (1-\beta)\overline{f}_k$ for some $\beta \in (0, 1)$.
(c) Set $X_k := \{x \in X : m_k(x) \le l_k\}$ and determine the new iterate by

$$x_{k+1} = \text{argmin}_{x \in X_k} \|x - x_k\|^2. \tag{1.8}$$

In the BL method, the localizer X_k is used to approximate the level set $L_k := \{x : f(x) \le l_k\}$, because the projection over L_k is often too difficult to compute. Intuitively, as k increases, the value of l_k will converge to f_X^*, and consequently both L_k and X_k will converge to the set of optimal solutions to problem (1.4). It is shown in [20] that the number of BL iterations required to find an ϵ-solution to problem (1.4), i.e., a point $\hat{x} \in X$ s.t. $f(\hat{x}) - f_X^* \le \epsilon$, can be bounded by $\mathcal{O}(1/\epsilon^2)$, which is optimal for general nonsmooth convex optimization in the black-box model.

Observe that for the above BL methods, the localizer X_k accumulates constraints, and hence the subproblem in Step c) becomes more and more expensive to solve. In order to overcome this difficulty, some restricted memory BL algorithms have been developed in [19,22]. In particular, Ben-Tal and Nemirovski [22] introduced the non-Euclidean restricted memory level (NERML) method, in which the number of extra linear constraints in X_k can be as small as 1 or 2, without affecting the optimal iteration complexity. Moreover, the objective function $\|\cdot\|^2$ in (1.8) is replaced by a general Bregman distance $d(\cdot)$ for exploiting the geometry of the feasible set X. From our understanding, the efficiency of NERML can be attributed to its combination of previous progresses on BL methods (e.g., [7,11,5,36]) with the incorporation of the mirror descent idea in order to exploit the geometry of the feasible set. Some more recent development of inexact proximal bundle methods and BL methods could be found in [23–30].

While the classic BL method was optimal for solving nonsmooth CP problems only, Lan [1] recently significantly generalized this method so that it can optimally solve any black-box CP problems, including nonsmooth, smooth and weakly smooth CP problems. In particular, for problem (1.4) over compact feasible set X, the two new BL methods proposed in [1], i.e., the accelerated bundle-level (ABL) and accelerated prox-level (APL) methods, can solve these problems optimally without requiring any information on problem parameters. The ABL method can be viewed as an accelerated version of the classic BL method. Same as the classic BL method, the lower bound on f_X^* is estimated from the cutting plane model m_k in (1.5), the upper bound on f_X^* is given by the best objective value found so far. The novelty of the ABL method exists in that three different sequences, i.e., $\{x_k^l\}$, $\{x_k\}$ and $\{x_k^u\}$, are used for updating the lower bound, prox-center, and upper bound respectively, which leads to its accelerated iteration complexity for smooth and weakly smooth problems. The APL method is a more practical, restricted memory version of the ABL method, which also employs non-Euclidean prox-functions to explore the geometry of the feasible set X. It is shown in [1] that both the ABL and APL methods achieve the optimal iteration complexity uniformly for smooth, weakly smooth and nonsmooth convex functions for solving problem (1.4). Moreover, by incorporating Nesterov's smoothing technique [5] into the APL method, Lan also presented in [1] that the uniform smoothing level (USL)

method which can achieve the optimal complexity for solving an important class of nonsmooth structured saddle point (SP) problems without requiring input of any problem parameters (see Sect. 2.2 for more details).

1.2 Motivation and contribution of this paper

One crucial problem associated with most existing BL type methods, including APL and USL, is that each iteration of these algorithms involves solving two optimization problems: first a linear programming problem to compute the lower bound, and then a constrained quadratic programming problem to update the prox-center or new iterate. In fact, the efficiency of these algorithms relies on the solutions to these two involved subproblems, and the latter one is often more complicated than the projection subproblem in the gradient projection type methods. Moreover, most existing BL type methods require the assumption that the feasible set is bounded due to the following two reasons. Firstly, the feasible set has to be bounded to compute a meaningful lower bound by solving the aforementioned linear programming problem. Secondly, the iteration complexity analysis of those methods, such as the classical BL method [20], NERML [22], ABL, APL and USL [1], relies on the assumption that the feasible set is compact. It should be noted that there exist some variants of BL methods [23,29,31,32] for solving nonsmooth CP problems in which the computation of the subproblem for updating the lower bound is skipped, so that the feasible set X is allowed to be unbounded. For instance, the level bundle method in [32] updates the level parameter directly when the distance from the stability center to the newly generated iterate becomes larger than a chosen parameter. However, all these methods are focusing on solving nonsmooth CP problems. From the complexity analysis point of view, when applied to solve smooth CP problems, the methods do not necessarily achieve the optimal rate of convergence for smooth convex optimization. In this paper, our main focus is to develop a BL type method that achieves the optimal rate of convergence for unconstrained smooth convex optimization. Our contribution in this paper mainly consists of the following three aspects.

Firstly, we propose the FAPL and FUSL methods that greatly reduce the computational cost per iteration of the APL and USL methods for solving ball-constrained CP problems. The improvement is achieved mainly by eliminating the linear optimization subproblem for computing the lower bound and removing the ball constraint in the quadratic subproblem by properly choosing the prox-functions. More importantly, we are able to show that with these simplifications the FAPL and FUSL methods can also achieve the optimal iteration complexity uniformly for smooth, nonsmooth and weakly smooth functions.

Secondly, we propose a novel algorithmic framework for solving unconstrained CP problems. The proposed framework solves unconstrained CP problems through solutions to a series of ball-constrained CP problems and achieves the same order of the iteration complexity as the corresponding ball-constrained CP problems. In particular, if there exists a uniformly optimal method (e.g., the APL and USL methods) that solves ball-constrained black-box or structured CP problems, then the proposed algorithm solves unconstrained black-box or structured CP problems with optimal complexity

without requiring the input of any additional problem parameters. To the best of our knowledge, this is the first time in the literature that the complexity analysis has been performed for BL type methods to solve unconstrained CP problems (see Sections 3.2 and 3.3 in [33] for more details).

Finally, we apply the FAPL and FUSL methods to solve large-scale least squares problems and total variation based image reconstruction problems. Our experimental results show that these algorithms outperform APL, NERML, Nesterov's optimal method [5], the accelerated primal dual (APD) method [8], Nesterov's smoothing (NEST-S) gradient method [5,34] and the accelerated linearized ADMM (AL-ADMM) with line-search method [15], and the MATLAB solver for linear systems, especially when the dimension and/or the Lipschitz constant of the problem is large. Moreover, by using the FAPL and FUSL methods, we could achieve more accurate solutions to the corresponding CP problems, which results in better reconstructed image qualities and lower acquisition rates.

1.3 Organization of the paper

The paper is organized as follows. In Sect. 2, the new FAPL and FUSL methods are proposed followed by their convergence analysis, then an exact approach is introduced to solve the subproblem in these algorithms. In Sect. 3, we present a general scheme to extend the optimal methods for ball-constrained CP problems to unconstrained CP problems. The applications to quadratic programming and image processing problems are presented in Sect. 4.

2 Fast prox-level type methods for ball-constrained problems

In this section, we discuss the following ball-constrained CP problem:

$$f^*_{\bar{x},R} := \min_{x \in B(\bar{x},R)} f(x), \tag{2.1}$$

where $B(\bar{x}, R) := \{x \in \mathbb{R}^n : \|x - \bar{x}\| \leq R\}$ denotes the closed Euclidean ball centered at $\bar{x}$ with radius R, and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function satisfying

$$f(y) - f(x) - \langle f'(x), y - x \rangle \leq \frac{M}{1+\rho}\|y - x\|^{1+\rho}, \quad \forall x, y \in B(\bar{x}, R), \tag{2.2}$$

for some $M > 0$ and $\rho \in [0, 1]$. This $f(\cdot)$ can be nonsmooth ($\rho = 0$), smooth ($\rho = 1$) and weakly smooth ($0 < \rho < 1$).

This section contains four subsections. We first present a much simplified APL method, referred to the fast APL (FAPL) method, for solving ball-constrained black-box CP problems in Sect. 2.1, and then present the fast USL (FUSL) method for solving a special class of ball-constrained structured CP problems in Sect. 2.2. We show how to solve the subproblems in these two algorithms in "Appendix A".

2.1 FAPL for ball-constrained black-box problems

Our goal in this subsection is to present the FAPL method, which can reduce the iteration cost for the APL method applied to problem (2.1). In particular, we show that only one subproblem, rather than two subproblems (see the two subproblems in Equations (2.9) and (2.10) in [1]) as in the APL method, is required in the FAPL method for defining a new iterate (or prox-center) and updating lower bound. We also demonstrate that the ball constraint in (2.1) can be eliminated from the subproblem by properly specifying the prox-function.

Similarly to the APL method, the FAPL method consists of outer-inner loops, and in each outer iteration, an inner loop, the FAPL gap reduction procedure denoted by $\mathcal{G}_{FAPL}$, is called to reduce the gap between the upper and lower bounds on $f^*_{\bar{x},R}$ in (2.1) by a constant factor.

We start by describing the FAPL gap reduction procedure in Procedure 1. This procedure differs from the gap reduction procedure used in the APL method in the following several aspects. Firstly, the localizers $\underline{Q}_k$ and $\overline{Q}_k$ in procedure $\mathcal{G}_{FAPL}$ (see Steps 1 and 4) only contain linear constraints and hence are possibly unbounded, while the localizers in the APL method must be compact. Secondly, we eliminate the subproblem that updates the lower bound on $f^*_{\bar{x},R}$ in the APL method. Instead, in the FAPL method, the lower bound is updated to the level l directly whenever $\underline{Q}_k = \emptyset$ or $\|x_k - \bar{x}\| > R$. Note that this strategy has been used in some existing BL methods [32]. Thirdly, we choose a specific prox-function $d(x) = \frac{1}{2}\|x - \bar{x}\|^2$, and as a result, all the three sequences $\{x_k\}$, $\{x^l_k\}$ and $\{x^u_k\}$ will reside in the ball $B(\bar{x}, R)$. At last, as we will show in next subsection, since the subproblem (2.6) only contains a limited number of linear constraints (depth of memory), we can solve it efficiently, or even exactly if the depth of memory is small (e.g., less than 10).

We now add a few more remarks about the technical details of Procedure 1. Firstly, Procedure 1 is terminated at Step 2 if $\underline{Q}_k = \emptyset$ or $\|x_k - \bar{x}\| > R$, which can be checked automatically when solving the subproblem (2.6) (see Subsection A for more details). Secondly, the set $\underline{Q}_k \neq \emptyset$ in Step 4 (otherwise, the procedure stops at Step 2). Moreover, we can show that $\underline{Q}_k$ is a subset of $\bar{Q}_k$ (see Lemma 2.2). Therefore, $\bar{Q}_k$ will never be empty at Step 4. Thirdly, in Step 4, Q_k can be any polyhedral set between $\underline{Q}_k$ and $\overline{Q}_k$. In practice, we can simply choose Q_k to be the intersection of the half-space $\{x \in \mathbb{R}^n : \langle x_k - \bar{x}, x - x_k\rangle \geq 0\}$ and a few most recently generated half-spaces, each of which is defined by $\{x \in \mathbb{R}^n : h(x^l_\tau, x) \leq l\}$ for some $1 \leq \tau \leq k$. Finally, in order to guarantee the termination of procedure $\mathcal{G}_{FAPL}$ and the optimal iteration complexity, the parameters $\{\alpha_k\}$ used in this procedure need to be properly chosen. One set of conditions that $\{\alpha_k\}$ should satisfy to guarantee the convergence of procedure $\mathcal{G}_{FAPL}$ is given as follows:

$$\alpha_1 = 1,\ 0 < \alpha_k \leq 1, \alpha_k \leq \frac{c}{k} \text{ and } \frac{1-\alpha_{k+1}}{\alpha^{1+\rho}_{k+1}} \leq \frac{1}{\alpha^{1+\rho}_k}, \forall k \geq 1, \tag{2.10}$$

for some constants $c > 0$ and $\forall \rho \in [0, 1]$.

The following lemma provides two examples for the selection of $\{\alpha_k\}$.

Procedure 1 The FAPL gap reduction procedure: $(x^+, \mathrm{lb}^+) = \mathcal{G}_{FAPL}(\hat{x}, \mathrm{lb}, R, \overline{x}, \beta, \theta)$

0: Set $k = 1$, $\overline{f}_0 = f(\hat{x})$, $l = \beta \cdot \mathrm{lb} + (1-\beta)\overline{f}_0$, $Q_0 = \mathbb{R}^n$, and let $x_0^u = \hat{x}$, $x_0 = \overline{x}$.

1: Update the cutting plane model:

$$x_k^l = (1-\alpha_k)x_{k-1}^u + \alpha_k x_{k-1}, \tag{2.3}$$

$$h(x_k^l, x) = f(x_k^l) + \left\langle f'(x_k^l), x - x_k^l \right\rangle, \tag{2.4}$$

$$\underline{Q}_k = \{x \in Q_{k-1} : h(x_k^l, x) \le l\}. \tag{2.5}$$

2: Update the prox-center and lower bound:

$$x_k = \mathrm{argmin}_{x \in \underline{Q}_k} \left\{ d(x) := \frac{1}{2}\|x - \overline{x}\|^2 \right\}. \tag{2.6}$$

If $\underline{Q}_k = \emptyset$ or $\|x_k - \overline{x}\| > R$, then **terminate** with outputs $x^+ = x_{k-1}^u$, $\mathrm{lb}^+ = l$.

3: Update the upper bound: set

$$\tilde{x}_k^u = (1-\alpha_k)x_{k-1}^u + \alpha_k x_k, \tag{2.7}$$

$$x_k^u = \begin{cases} \tilde{x}_k^u, & \text{if } f(\tilde{x}_k^u) < \overline{f}_{k-1}, \\ x_{k-1}^u, & \text{otherwise,} \end{cases} \tag{2.8}$$

and $\overline{f}_k = f(x_k^u)$. If $\overline{f}_k \le l + \theta(\overline{f}_0 - l)$, then **terminate** with $x^+ = x_k^u$, $\mathrm{lb}^+ = \mathrm{lb}$.

4: Choose any polyhedral set Q_k satisfying $\underline{Q}_k \subseteq Q_k \subseteq \overline{Q}_k$, where

$$\overline{Q}_k := \{x \in \mathbb{R}^n : \langle x_k - \overline{x}, x - x_k \rangle \ge 0\}. \tag{2.9}$$

Set $k = k + 1$ and go to Step 1.

Lemma 2.1 (a) *If $\alpha_k = 2/(k+1)$, $k = 1, 2, \ldots$, then the condition (2.10) is satisfied with $c = 2$.*

(b) *If $\{\alpha_k\}$ is recursively defined by*

$$\alpha_1 = 1, \alpha_{k+1} = \frac{-\alpha_k^2 + \sqrt{\alpha_k^4 + 4\alpha_k^2}}{2}, \forall k \ge 1, \tag{2.11}$$

then the condition (2.10) holds with $c = 2$. Note that the stepsizes in (2.11) satisfies

$$\alpha_{k+1}^2 = (1-\alpha_{k+1})\alpha_k^2 \tag{2.12}$$

Proof To prove part a), note that if $\alpha_k = 2/(k+1)$, then we have $\alpha_1 = 1$, $0 < \alpha_k \le 1$, and $\alpha_k \le 2/k$. Moreover, for any $\rho \in [0, 1]$ we have

$$\begin{aligned}(1-\alpha_{k+1}) \cdot \frac{\alpha_k^{1+\rho}}{\alpha_{k+1}^{1+\rho}} &= \frac{k}{k+2} \cdot \left(\frac{k+2}{k+1}\right)^{1+\rho} = \frac{k}{k+1} \cdot \left(\frac{k+2}{k+1}\right)^{\rho} \\ &\le \frac{k}{k+1} \cdot \left(\frac{k+1}{k}\right)^{\rho} = \left(\frac{k}{k+1}\right)^{1-\rho} \le 1.\end{aligned}$$

Therefore (2.10) holds.

We continue to prove part b). First it is easy to verify that the stepsizes defined in (2.11) satisfy the relation (2.12). Moreover, by (2.12) we also have $\alpha_{k+1} \neq 0$ as long as $\alpha_k \neq 0$. Therefore, noting that $\alpha_1 = 1$ we have $\alpha_k \neq 0$ for all $k \geq 1$. Consequently, from (2.11) we have

$$\alpha_{k+1} = \frac{4\alpha_k^2}{2(\alpha_k^2 + \sqrt{\alpha_k^4 + 4\alpha_k^2})} \leq \frac{4\alpha_k^2}{4\alpha_k^2} = 1, \ \forall k \geq 1,$$

and thus we have $\alpha_k \in (0, 1]$ for all $k \geq 1$. Using this result and (2.12), we also have $\alpha_{k+1}^2 < \alpha_k^2$, hence $\alpha_{k+1} < \alpha_k$. Now rearranging the terms in (2.12), we observe that $\alpha_k^2 - \alpha_{k+1}^2 = \alpha_{k+1}\alpha_k^2$. Using this observation and the previous result that $\alpha_{k+1} < \alpha_k$, we obtain

$$\frac{1}{\alpha_{k+1}} - \frac{1}{\alpha_k} = \frac{\alpha_k - \alpha_{k+1}}{\alpha_k \alpha_{k+1}} = \frac{\alpha_k^2 - \alpha_{k+1}^2}{\alpha_{k+1}\alpha_k^2} \cdot \frac{\alpha_k}{\alpha_k + \alpha_{k+1}} = \frac{\alpha_k}{\alpha_k + \alpha_{k+1}} > \frac{1}{2}, \forall k \geq 1.$$

Using the above inequality and noting that $\alpha_1 = 1$, we have

$$\begin{aligned} \frac{1}{\alpha_k} &= \left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}}\right) + \left(\frac{1}{\alpha_{k-1}} - \frac{1}{\alpha_{k-2}}\right) + \cdots \left(\frac{1}{\alpha_2} - \frac{1}{\alpha_1}\right) \\ &\quad + \frac{1}{\alpha_1} > (k-1) \cdot \frac{1}{2} + 1 = \frac{k+1}{2}, \end{aligned} \tag{2.13}$$

and hence $\alpha_k \leq 2(k+1) < 2/k$. Finally, using (2.12) and the result that $\alpha_{k+1} < \alpha_k$, for any $\rho \in [0, 1]$ we have

$$(1 - \alpha_{k+1}) \cdot \frac{\alpha_k^{1+\rho}}{\alpha_{k+1}^{1+\rho}} = \frac{(1 - \alpha_{k+1})\alpha_k^2}{\alpha_{k+1}^2} \cdot \frac{\alpha_k^{\rho-1}}{\alpha_{k+1}^{\rho-1}} = \frac{\alpha_k^{\rho-1}}{\alpha_{k+1}^{\rho-1}} = \left(\frac{\alpha_{k+1}}{\alpha_k}\right)^{1-\rho} < 1.$$

Therefore (2.11) holds. □

The following lemma describes some important observations regarding the execution of procedure $\mathcal{G}_{FAPL}$.

Lemma 2.2 *Let $\mathcal{E}_f(l) := \{x \in B(\overline{x}, R) : f(x) \leq l\}$, the following statements hold for procedure $\mathcal{G}_{FAPL}$.*

(a) *If $\mathcal{E}_f(l) \neq \emptyset$, then $\mathcal{E}_f(l) \subseteq \underline{Q}_k \subseteq Q_k \subseteq \overline{Q}_k$ for any $k \geq 1$.*
(b) *If $\underline{Q}_k \neq \emptyset$, then problem (2.6) in Step 2 has a unique solution. Moreover, if procedure $\mathcal{G}_{FAPL}$ terminates at Step 2, then $l \leq f^*_{\overline{x},R}$.*

Proof To prove part a), it suffices to prove that $\mathcal{E}_f(l) \subseteq \underline{Q}_k$ and $\underline{Q}_k \subseteq \overline{Q}_k$ for all $k \geq 1$, since from the definition of Q_k we already have $\underline{Q}_k \subseteq Q_k \subseteq \overline{Q}_k$. we first use induction to prove the former relation. As Q_0 is set to $\mathbb{R}^n$, we have $\mathcal{E}_f(l) \subseteq Q_0$. Moreover, if $\mathcal{E}_f(l) \subseteq Q_{k-1}$ for some $k \geq 1$, then from the definition of $\underline{Q}_k$ in (2.5)

and the observation that $h(x_k^l, z) \leq f(z) \leq l$ for all $z \in \mathcal{E}_f(l)$, we have $\mathcal{E}_f(l) \subseteq \underline{Q}_k$. To prove the relation that $\underline{Q}_k \subseteq \overline{Q}_k$, observe that the definition of $\overline{Q}_k$ in (2.9) implies that $\overline{Q}_k = \{x \in \mathbb{R}^n : d(x) \geq d(x_k)\}$. Combining such observation with the definition of x_k in (2.6), we have $\underline{Q}_k \subseteq \overline{Q}_k$ and conclude part a).

We now provide the proof of part b). From the definition of Q_k in Step 4 and the definition of $\underline{Q}_k$ in (2.5), we can see that $\underline{Q}_k$ is the intersection of half-spaces, hence it is convex and closed. Therefore, the subproblem (2.6) always has a unique solution as long as $\underline{Q}_k$ is non-empty.

To finish the proof it suffices to show that $\mathcal{E}_f(l) = \emptyset$ when either $\underline{Q}_k = \emptyset$ or $\|x_k - \overline{x}\| > R$, which can be proved by contradiction. Firstly, if $\underline{Q}_k = \emptyset$ but $\mathcal{E}_f(l) \neq \emptyset$, then by part a) proved above, we have $\mathcal{E}_f(l) \subseteq \underline{Q}_k$, which contradicts the assumption that $\underline{Q}_k$ is empty. On the other hand, suppose that $\|x_k - \overline{x}\| > R$ and $\mathcal{E}_f(l) \neq \emptyset$, let $x_R^* \in \operatorname{Argmin}_{x \in B(\overline{x}, R)} f(x)$, it is clear that $x_R^* \in \mathcal{E}_f(l) \subseteq \underline{Q}_k$ by a), however $\|x_R^* - \overline{x}\| \leq R < \|x_k - \overline{x}\|$ which contradicts the definition of x_k in (2.6). □

Let $\text{ub} := f(\hat{x})$, $\text{ub}^+ := f(x^+)$ be the input and output upper bounds on $f^*_{\overline{x},R}$ in procedure $\mathcal{G}_{FAPL}$, respectively. The following lemma shows that whenever procedure $\mathcal{G}_{FAPL}$ terminates, the gap between the upper and lower bounds on $f^*_{\overline{x},R}$ is reduced by a constant factor.

Lemma 2.3 *Whenever procedure $\mathcal{G}_{FAPL}$ terminates, we have* $\text{ub}^+ - \text{lb}^+ \leq q(\text{ub} - \text{lb})$, *where*

$$q := \max\{\beta, 1 - (1-\theta)\beta\}. \tag{2.14}$$

Proof By the definition of x_k^u in (2.7) and the definition of $\overline{f}_k$ in Step 3 of procedure $\mathcal{G}_{FAPL}$, we have $\overline{f}_k \leq \overline{f}_{k-1}$, $\forall k \geq 1$, which implies $\text{ub}^+ \leq \text{ub}$. Procedure $\mathcal{G}_{FAPL}$ could terminate at either Step 2 or Step 3. We first assume that it terminates at Step 2 at K^{th} iteration. The termination condition gives $\text{lb}^+ = l = \beta \cdot \text{lb} + (1-\beta)\text{ub}$, then we have

$$\text{ub}^+ - \text{lb}^+ \leq \text{ub} - \beta\,\text{lb} - (1-\beta)\text{ub} = \beta(\text{ub} - \text{lb}).$$

Next suppose that Procedure 1 terminates at Step 3 at K^{th} iteration. We have $\text{ub}^+ = f(x^+) = \overline{f}_K \leq l + \theta(\text{ub} - l)$, since $\text{lb}^+ \geq \text{lb}$ and $l = \beta \cdot \text{lb} + (1-\beta)\text{ub}$, we conclude that

$$\text{ub}^+ - \text{lb}^+ \leq l + \theta\,(\text{ub} - l) - \text{lb} = [1 - (1-\theta)\beta](\text{ub} - \text{lb}).$$

The lemma is proved by combining the above two cases. □

We now provide a bound on the number of iterations performed by procedure $\mathcal{G}_{FAPL}$. Note that the proof of this result is similar to Theorem 3 in [1].

Proposition 2.4 *If the stepsizes $\{\alpha_k\}_{k \geq 1}$ are chosen such that (2.10) holds, then the number of iterations performed by procedure $\mathcal{G}_{FAPL}$ does not exceed*

$$N(\Delta) := \left(\frac{c^{1+\rho} M R^{1+\rho}}{(1+\rho)\theta\beta\Delta} \right)^{\frac{2}{1+3\rho}} + 1, \tag{2.15}$$

where $\Delta := \text{ub} - \text{lb}$.

Proof Suppose that procedure $\mathcal{G}_{FAPL}$ does not terminate at the K^{th} iteration for some $K > 0$. Then observing that $x_k \in Q_k \subseteq Q_{k-1} \subseteq \overline{Q}_{k-1}$ for any $2 \le k \le K$ due to (2.5) and (2.6), and $x_1 \in Q_0$, $x_0 = \mathrm{argmin}_{x \in Q_0} d(x)$, we have $\langle \nabla d(x_{k-1}), x_k - x_{k-1} \rangle \ge 0, \forall 1 \le k \le K$. Since $d(x)$ is strongly convex with modulus 1, we also have

$$d(x_k) \ge d(x_{k-1}) + \langle \nabla d(x_{k-1}), x_k - x_{k-1} \rangle + \frac{1}{2}\|x_k - x_{k-1}\|^2.$$

Combining the two relations above, it implies $\frac{1}{2}\|x_k - x_{k-1}\|^2 \le d(x_k) - d(x_{k-1})$. Summing up these inequalities for $1 \le k \le K$, we conclude that

$$\frac{1}{2}\sum_{k=1}^{K} \|x_k - x_{k-1}\|^2 \le d(x_K) = \frac{1}{2}\|x_K - \overline{x}\|^2 \le \frac{1}{2}R^2. \tag{2.16}$$

By (2.2), (2.7), (2.8) and the convexity of f, we have for any $1 \le k \le K$,

$$f(x_k^u) \le f(\tilde{x}_k^u) \le h(x_k^l, \tilde{x}_k^u) + \frac{M}{1+\rho}\|\tilde{x}_k^u - x_k^l\|^{1+\rho} \tag{2.17}$$

$$= (1-\alpha_k)h(x_k^l, x_{k-1}^u) + \alpha_k h(x_k^l, x_k) + \frac{M}{1+\rho}\|\tilde{x}_k^u - x_k^l\|^{1+\rho}. \tag{2.18}$$

Since $h(x_k^l, x_{k-1}^u) \le f(x_{k-1}^u)$, $h(x_k^l, x_k) \le l$ due to (2.5) and (2.6), and $\tilde{x}_k^u - x_k^l = \alpha_k(x_k - x_{k-1})$ due to (2.3) and (2.7), we have

$$f(x_k^u) - l \le (1-\alpha_k)(f(x_{k-1}^u) - l) + \frac{\alpha_k^{1+\rho} M}{1+\rho}\|x_k - x_{k-1}\|^{1+\rho} \tag{2.19}$$

Dividing both sides of (2.19) by $\alpha_k^{1+\rho}$, and then summing up these inequalities for $1 \le k \le K$, by (2.10) and the fact $f(x_k^u) - l \ge 0, \forall 1 \le k \le K$, we have

$$f(x_K^u) - l \le \frac{\alpha_K^{1+\rho} M}{1+\rho}\sum_{k=1}^{K} \|x_k - x_{k-1}\|^{1+\rho} \tag{2.20}$$

Apply Hölder's inequality, and use (2.16), we have

$$\sum_{k=1}^{K} \|x_k - x_{k-1}\|^{1+\rho} \le K^{\frac{1-\rho}{2}} \left(\sum_{k=1}^{K} \|x_k - x_{k-1}\|^2\right)^{\frac{1+\rho}{2}} \le K^{\frac{1-\rho}{2}} R^{1+\rho},$$

and $a_K^{1+\rho} \le c^{1+\rho} K^{-(1+\rho)}$ due to (2.10). Therefore (2.20) gives

$$f(x_K^u) - l \le \frac{M R^{1+\rho}}{(1+\rho)} \cdot \frac{c^{1+\rho}}{K^{\frac{1+3\rho}{2}}} \tag{2.21}$$

In view of Step 3 in procedure 1, and using the fact that $l = \beta \cdot \text{lb} + (1-\beta)\text{ub}$ in Step 0, we also have

$$f(x_K^u) - l > \theta(\text{ub} - l) = \theta\beta\Delta.$$

Combining the above two relations, and using (2.10) and (2.16), we obtain

$$\theta\beta\Delta < \frac{MR^{1+\rho}}{(1+\rho)} \cdot \frac{c^{1+\rho}}{K^{\frac{1+3\rho}{2}}}, \tag{2.22}$$

which implies that

$$K < \left(\frac{c^{1+\rho} M R^{1+\rho}}{(1+\rho)\theta\beta\Delta} \right)^{\frac{2}{1+3\rho}}. \tag{2.23}$$

□

In view of Lemma 2.3 and Proposition 2.4, we are now ready to describe the FAPL method, which performs a sequence of calls to procedure $\mathcal{G}_{FAPL}$ until an approximate solution with sufficient accuracy is found.

Algorithm 1 The fast accelerated prox-level (FAPL) method

0: Given ball $B(\overline{x}, R)$, choose initial point $p_0 \in B(\overline{x}, R)$, tolerance $\epsilon > 0$ and parameters $\beta, \theta \in (0, 1)$.
1: Set $p_1 \in \text{Argmin}_{x \in B(\overline{x}, R)} h(p_0, x)$, $\text{lb}_1 = h(p_0, p_1)$, $\text{ub}_1 = \min\{f(p_0), f(p_1)\}$, let $\hat{x}_1$ be either p_0 or p_1 such that $f(\hat{x}_1) = \text{ub}_1$, and $s = 1$.
2: If $\text{ub}_s - \text{lb}_s \le \epsilon$, **terminate** and **output** approximate solution $\hat{x}_s$.
3: Set $(\hat{x}_{s+1}, \text{lb}_{s+1}) = \mathcal{G}_{FAPL}(\hat{x}_s, \text{lb}_s, R, \overline{x}, \beta, \theta)$ and $\text{ub}_{s+1} = f(\hat{x}_{s+1})$.
4: Set $s = s + 1$ and go to Step 2.

For the sake of simplicity, each iteration of Procedure $\mathcal{G}_{FAPL}$ is also referred to as an inner iteration of the FAPL method. Our complexity bound is then built based on the total number of combined inner iterations of the FAPL method, namely, the number of inner iterations performed by all calls to Procedure $\mathcal{G}_{FAPL}$, combined together. The following theorem establishes the complexity bounds on the numbers of gap reduction procedures $\mathcal{G}_{FAPL}$ and total combined inner iterations performed by the FAPL method, its proof is similar to that of Theorem 4 in [1].

Theorem 2.5 *For any given $\epsilon > 0$, if the stepsizes $\{\alpha_k\}$ in procedure $\mathcal{G}_{FAPL}$ are chosen such that (2.10) holds, then the following statements hold for the FAPL method to compute an ϵ-solution to problem (2.1).*

(a) *The number of gap reduction procedures $\mathcal{G}_{FAPL}$ performed by the FAPL method does not exceed*

$$S := \left\lceil \max\left\{0, \log_{\frac{1}{q}} \left(\frac{(2R)^{1+\rho} M}{(1+\rho)\epsilon} \right)\right\} \right\rceil. \tag{2.24}$$

(b) *The total number of combined inner iterations performed by the FAPL method can be bounded by*

$$N(\epsilon) := S + \frac{1}{1 - q^{\frac{2}{1+3\rho}}} \left(\frac{c^{1+\rho} M R^{1+\rho}}{(1+\rho)\theta\beta\epsilon} \right)^{\frac{2}{1+3\rho}}, \tag{2.25}$$

where q is defined in (2.14).

Proof We first prove part a). Let $\Delta_s := \mathrm{ub}_s - \mathrm{lb}_s$, without loss of generality, we assume that $\Delta_1 > \epsilon$. In view of Step 0 in Algorithm 1 and (2.2), we have

$$\Delta_1 \le f(p_1) - h(p_0, p_1) = f(p_1) - f(p_0) - \langle f'(p_0), p_1 - p_0 \rangle \le \frac{(2R)^{1+\rho} M}{1+\rho}. \tag{2.26}$$

Also, by Lemma 2.3 we can see that $\Delta_{s+1} \le q\Delta_s$ for any $s \ge 1$, which implies that

$$\Delta_{s+1} \le q^s \Delta_1, \ \forall s \ge 0.$$

Moreover, if an ϵ-solution is found after performing S gap reduction procedures $\mathcal{G}_{FAPL}$, then we have

$$\Delta_S > \epsilon \ge \Delta_{S+1}. \tag{2.27}$$

Combining the above three inequalities, we conclude that

$$\epsilon < q^{S-1} \Delta_1 \le q^{S-1} \frac{(2R)^{1+\rho} M}{1+\rho}, \tag{2.28}$$

and part a) follows immediately from the above inequality. We are now ready to prove part b). In view of Lemma 2.3 and (2.27), we have $\Delta_s \ge \epsilon q^{s-S}$ for any $1 \le s \le S$. Using this estimate, part a), and Proposition 2.4, we conclude that the total number of combined inner iterations performed by the FAPL method is bounded by

$$\begin{aligned} N(\epsilon) = \sum_{s=1}^{S} N_s &= S + \left(\frac{c^{1+\rho} M R^{1+\rho}}{(1+\rho)\theta\beta\epsilon} \right)^{\frac{2}{1+3\rho}} \sum_{s=1}^{S} q^{\frac{2(S-s)}{1+3\rho}} \\ &< S + \frac{1}{1 - q^{\frac{2}{1+3\rho}}} \left(\frac{c^{1+\rho} M R^{1+\rho}}{(1+\rho)\theta\beta\epsilon} \right)^{\frac{2}{1+3\rho}}, \end{aligned} \tag{2.29}$$

where N_s denotes the number of the inner iterations performed by the s*th* gap reduction procedure $\mathcal{G}_{FAPL}$ for any $1 \le s \le S$. □

A few remarks are in place for Theorem 2.5. First, the number of iterations performed by the FAPL method is defined as the number of gradient/subgradient evaluations, but the FAPL method requires two function evaluations for each iteration. Therefore, the number of function evaluations of the FAPL method is the same

Table 1 The total number of inner iterations $N(\epsilon)$ performed by the FAPL method with respect to different smoothness condition of f

	Nonsmooth ($\rho = 0$)	Smooth ($\rho = 1$)	Weakly smooth ($0 < \rho < 1$)
$N(\epsilon)$	$\mathcal{O}\left(\frac{M^2R^2}{\epsilon^2}\right)$	$\mathcal{O}\left(\sqrt{\frac{MR^2}{\epsilon}}\right)$	$\mathcal{O}\left(\left(\frac{MR^{1+\rho}}{\epsilon}\right)^{\frac{2}{1+3\rho}}\right)$

as that of the APL method in [1], but twice as some BL methods for nonsmooth optimization (e.g., [20,22]) and Nesterov's accelerate gradient method [5]. Second, when f is nonsmooth (i.e., $\rho = 0$ in (2.2)), setting $\rho = 0$ in (2.25) we have

$$N(\epsilon) = S + \frac{1}{1-q^2}\left(\frac{cMR}{\theta\beta\epsilon}\right)^2 = \mathcal{O}\left(\frac{M^2R^2}{\epsilon^2}\right). \tag{2.30}$$

Following similar computations, we can summarize the total number of inner iterations $N(\epsilon)$ performed by the FAPL method in Table 1. Third, it is important to note that, in the case when f is smooth, the total number of inner iterations $N(\epsilon)$ in Table 1 is smaller in an order than that of the case when f is nonsmooth. Finally, we can compare the iteration complexity results in Table 1 with other BL methods in the literature. For BL methods that focus on nonsmooth CP problems, e.g., the NERML method in [22], their complexity results matches our result for nonsmooth f (i.e., $\rho = 0$). However, we are able to improve the complexity results when f is smooth or weakly smooth. To the best of our knowledge, in the current literature the APL method in [1] is the only other method that is able to achieve all the three iteration complexity results in Table 1. However, the APL method is not applicable to unconstrained CP problems, while we can extend our study to an expansion algorithm for unconstrained CP problems (see Sect. 3 later).

2.2 FUSL for ball-constrained structured problems

In this subsection, we still consider the ball-constrained problem (2.1), but assume that its objective function is given by

$$f(x) := \hat{f}(x) + F(x), \tag{2.31}$$

where $\hat{f}$ is a smooth convex function, i.e., $\exists L_{\hat{f}} > 0$ s.t.

$$\hat{f}(y) - \hat{f}(x) - \langle \nabla \hat{f}(x), y - x\rangle \le \frac{L_{\hat{f}}}{2}\|y - x\|^2, \tag{2.32}$$

and

$$F(x) := \max_{y\in Y}\{\langle Ax, y\rangle - \hat{g}(y)\}. \tag{2.33}$$

Here, $Y \subseteq \mathbb{R}^m$ is a compact convex set, $\hat{g} := Y \to \mathbb{R}$ is a relatively simple convex function, and $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear operator. We will integrate Nesterov's smoothing technique for minimizing nonsmooth functions [5] into the FAPL method to solve problem (2.1)–(2.31) (i.e., problem (2.1) with f defined in (2.31)), which can reduce the iteration cost of the USL method [1].

Let $v : Y \to \mathbb{R}$ be a prox-function with modulus σ_v and denoting $c_v := \operatorname{argmin}_{v\in Y} v(y)$, by employing the idea in the important work of Nesterov [5], we can approximate F in (2.33) by the smooth function

$$F_\eta(x) := \max_{y\in Y}\{\langle Ax, y\rangle - \hat{g}(y) - \eta V(y)\}, \tag{2.34}$$

where $\eta > 0$ is called the smoothing parameter, and $V(\cdot)$ is the Bregman divergence defined by

$$V(y) := v(y) - v(c_v) - \langle \nabla v(c_v), y - c_v\rangle . \tag{2.35}$$

It was shown in [5] that the gradient of $F_\eta(\cdot)$ given by $\nabla F_\eta(x) = A^* y^*(x)$ is Lipschitz continuous with constant

$$L_\eta := \|A\|^2/(\eta\sigma_v), \tag{2.36}$$

where $\|A\|$ is the operator norm of A, A^* is the adjoint operator, and $y^*(x) \in Y$ is the solution to the optimization problem (2.34). Moreover, the "closeness" of $F_\eta(\cdot)$ to $F(\cdot)$ depends linearly on the smoothing parameter η, i.e.,

$$F_\eta(x) \le F(x) \le F_\eta(x) + \eta D_{v,Y},\ \forall x \in X, \tag{2.37}$$

where

$$D_{v,Y} := \max_{y,z\in Y}\left\{v(y) - v(z) - \langle \nabla v(z), y - z\rangle\right\}. \tag{2.38}$$

Therefore, if we denote

$$f_\eta(x) := \hat{f}(x) + F_\eta(x), \tag{2.39}$$

then

$$f_\eta(x) \le f(x) \le f_\eta(x) + \eta D_{v,Y}. \tag{2.40}$$

Applying an optimal gradient method to minimize the smooth function f_η in (2.39), Nesterov proves in [5] that the iteration complexity for computing an ϵ-solution to problem (2.1)–(2.31) is bounded by $\mathcal{O}(\frac{1}{\sqrt{\epsilon}} + \frac{1}{\epsilon})$. However, the values of quite a few problem parameters, such as $\|A\|, \sigma_v$ and $D_{v,Y}$, are required for the implementation of Nesterov's smoothing scheme.

By incorporating Nesterov's smoothing technique [5] into the APL method, Lan developed in [1] a new bundle-level type method, namely the uniform smoothing level (USL) method, to solve structured problems given in the form of (2.1)–(2.31). While the USL method achieves the same optimal iteration complexity as Nesterov's

smoothing scheme in [5], one advantage of the USL method over Nesterov's smoothing scheme is that the smoothing parameter η is adjusted dynamically during the execution, and an estimate of $D_{v,Y}$ is obtained automatically, which makes the USL method problem parameter free. However, similar to the APL method, each iteration of the USL method involves the solutions to two subproblems. Based on the USL method in [1] and our analysis of the FAPL method in Sect. 2.1, we propose a fast USL (FUSL) method that solves problem (2.1)–(2.31) with the same optimal iteration complexity as the USL method, but requiring only to solve one simpler subproblem in each iteration.

Similar to the FAPL method, the FUSL method has an outer-inner loops structure, and each outer iteration calls an inner loop, a gap reduction procedure denoted by $\mathcal{G}_{FUSL}$, to reduce the gap between the upper and lower bounds on $f^*_{\overline{x},R}$ in (2.1)–(2.31) by a constant factor unless an ϵ-solution is found. We start by describing procedure $\mathcal{G}_{FUSL}$.

Procedure 2 The FUSL gap reduction procedure: $(x^+, D^+, \mathrm{lb}^+) = \mathcal{G}_{FUSL}(\hat{x}, D, \mathrm{lb}, R, \overline{x}, \beta, \theta)$

0: Let $k = 1$, $\overline{f}_0 = f(\hat{x})$, $l = \beta \cdot \mathrm{lb} + (1-\beta)\overline{f}_0$, $Q_0 = \mathbb{R}^n$, $x^u_0 = \hat{x}$, $x_0 = \overline{x}$, and

$$\eta := \theta(\overline{f}_0 - l)/(2D). \tag{2.41}$$

1: Update the cutting plane model: Set x^l_k to (2.3), $\underline{Q}_k$ to (2.5), and

$$h(x^l_k, x) = h_\eta(x^l_k, x) = f_\eta(x^l_k) + \left\langle f'_\eta(x^l_k), x - x^l_k \right\rangle. \tag{2.42}$$

2: Update the prox-center: Set x_k to (2.6). If $\underline{Q}_k = \emptyset$ or $\|x_k - \overline{x}\| > R$, then **terminate** with outputs $x^+ = x^u_{k-1}$, $D^+ = D$, $\mathrm{lb}^+ = l$.

3: Update the upper bound and the estimate of $D_{v,Y}$: Set $\tilde{x}^u_k$ to (2.7), x^u_k to (2.8), and $\overline{f}_k = f(x^u_k)$. Check the following conditions:

3a) if $f(x^u_k) \le l + \theta(\overline{f}_0 - l)$, then **terminate** with outputs $x^+ = x^u_k$, $D^+ = D$, $\mathrm{lb}^+ = \mathrm{lb}$;

3b) if $f(x^u_k) > l + \theta(\overline{f}_0 - l)$ and $f_\eta(x^u_k) \le l + \frac{\theta}{2}(\overline{f}_0 - l)$, then **terminate** with outputs $x^+ = x^u_k$, $D^+ = 2D$, $\mathrm{lb}^+ = \mathrm{lb}$.

4: Choose Q_k as same as Step 4 in $\mathcal{G}_{FAPL}$. Set $k = k + 1$, and go to step 1.

A few remarks about procedure $\mathcal{G}_{FUSL}$ are in place. Firstly, since the nonsmooth objective function f is replaced by its smoothed approximation f_η, we replace the cutting plane model in (1.7) with the one for f_η (see (2.42)). Also note that for the USL method in [1], $\hat{f}$ is assumed to be a simple Lipschitz continuous convex function, and only F_η is approximated by the linear estimation. However, in the FUSL method, we assume $\hat{f}$ is general smooth and convex, and linearize both $\hat{f}$ and F_η in (2.42). Secondly, the smoothing parameter η is specified as a function of the parameter D, $\bar{f}_0$ and l, where D is an estimator of $D_{v,Y}$ defined by (2.38) and given as an input parameter for procedure $\mathcal{G}_{FUSL}$. Thirdly, same as the FAPL method, the parameters $\{\alpha_k\}$ are chosen according to (2.10). Such conditions are required to guarantee the optimal iteration complexity of the FUSL method for solving problem (2.1)–(2.31). Fourthly, similar to the FAPL method, the localizers $\underline{Q}_k$, Q_k, $\overline{Q}_k$ only contain a limited

number of linear constraints, and there is only one subproblem (i.e., (2.6)) involved in procedure $\mathcal{G}_{FUSL}$, which can be solved exactly when the depth of memory is small.

The following lemma provides some important observations about procedure $\mathcal{G}_{FUSL}$, which are similar to those for the USL gap reduction procedure in [1].

Lemma 2.6 *The following statements hold for procedure* $\mathcal{G}_{FUSL}$.

(a) *If procedure* $\mathcal{G}_{FUSL}$ *terminates at Steps* 2 *or* 3*(a), then we have* $\mathrm{ub}^+ - \mathrm{lb}^+ \le q(\mathrm{ub} - \mathrm{lb})$, *where* q *is defined by* (2.14) *and* $\mathrm{ub} := f(\hat{x})$, $\mathrm{ub}^+ := f(x^+)$.
(b) *If procedure* $\mathcal{G}_{FUSL}$ *terminates at Step 3b), then* $D < D_{v,Y}$ *and* $D^+ < 2D_{v,Y}$.

Proof The proof of part (a) is the same as that of Lemma 2.3, and we only prove part (b) here. By the termination condition at Step 3b), we have

$$f(x_k^u) - f_\eta(x_k^u) > \frac{\theta}{2}(\overline{f}_0 - l).$$

We conclude from the above relation, (2.40), and (2.41) that

$$D_{v,Y} \ge \frac{f(x_k^u) - f_\eta(x_k^u)}{\eta} > \frac{\theta(\overline{f}_0 - l)}{2\eta} = D.$$

Finally, $D^+ < 2D_{v,Y}$ follows immediately from the above relation and the definition of D^+ in Step 3b). □

The following result provides a bound on the number of iterations performed by procedure $\mathcal{G}_{FUSL}$.

Proposition 2.7 *Suppose that* $\{\alpha_k\}_{k\ge 1}$ *in procedure* $\mathcal{G}_{FUSL}$ *are chosen such that* (2.10) *holds. Then, the number of iterations performed by this procedure does not exceed*

$$\overline{N}(\Delta, D) := cR\left(\sqrt{\frac{L_{\hat{f}}}{\theta\beta\Delta}} + \frac{\sqrt{2}\|A\|}{\theta\beta\Delta}\sqrt{\frac{D}{\sigma_v}}\right) + 1, \tag{2.43}$$

where $\Delta := f(\hat{x}) - \mathrm{lb}$.

Proof It is easy to see that the gradient of f_η in (2.31) has Lipschitz continuous gradient with constant $L = L_{\hat{f}} + L_\eta$, where L_η and $L_{\hat{f}}$ are defined in (2.36) and (2.32), respectively. Suppose that procedure $\mathcal{G}_{FUSL}$ does not terminate at iteration K. As the $\mathcal{G}_{FUSL}$ procedure could be viewed as applying the $\mathcal{G}_{FAPL}$ procedure to f_η, similar to the discussion on (2.21), and notice that $\rho = 1$ as f_η is smooth, we have

$$f_\eta(x_K^u) - l \le \frac{c^2 L d(x_K)}{K^2} \le \frac{c^2 L R^2}{2K^2}, \tag{2.44}$$

where c is defined in (2.10). Also, since procedure $\mathcal{G}_{FUSL}$ does not terminate at iteration K, in view of the termination condition in Step 3b) and the definition of l in Step 0, we have

$$f_\eta(x_K^u) - l > \frac{\theta\beta\Delta}{2}. \tag{2.45}$$

By (2.44), (2.45), (2.36) and (2.41), we conclude that

$$K < \sqrt{\frac{c^2 L R^2}{\theta\beta\Delta}} \le cR\left(\sqrt{\frac{L_{\hat{f}}}{\theta\beta\Delta}} + \frac{\sqrt{2}\|A\|}{\theta\beta\Delta}\sqrt{\frac{D}{\sigma_v}}\right). \tag{2.46}$$

□

We are now ready to describe the FUSL method which iteratively calls procedure $\mathcal{G}_{FUSL}$ to solve the structured saddle point problem (2.1)-(2.31).

Algorithm 2 The fast uniform smoothing level (FUSL) method

0: Given ball $B(\overline{x}, R)$, choose initial point $p_0 \in B(\overline{x}, R)$, prox-function $v(\cdot)$ for the smoothing function F_η in (2.34) and (2.35), initial guess D_1 on the size $D_{v,Y}$ in (2.38), tolerance $\varepsilon > 0$, and parameters $\beta, \theta \in (0, 1)$.
1: Set $p_1 \in \mathrm{Argmin}_{x\in B(\overline{x},R)} h(p_0, x)$, $\mathrm{lb}_1 = h(p_0, p_1)$, $\mathrm{ub}_1 = \min\{f(p_0), f(p_1)\}$, let $\hat{x}_1$ be either p_0 or p_1 such that $f(\hat{x}_1) = \mathrm{ub}_1$, and $s = 1$.
2: If $\mathrm{ub}_s - \mathrm{lb}_s \le \epsilon$, **terminate** and **output** approximate solution $\hat{x}$.
3: Set $(\hat{x}_{s+1}, D_{s+1}, \mathrm{lb}_{s+1}) = \mathcal{G}_{FUSL}(\hat{x}_s, D_s, \mathrm{lb}_s, R, \overline{x}, \beta, \theta)$ and $\mathrm{ub}_{s+1} = f(\hat{x})$.
4: Set $s = s + 1$ and go to Step 2.

For simplicity, we say that a phase (i.e., an outer iteration) of the FUSL method occurs when s increases by 1. More specifically, similar to the USL method, we classify two types of phases in the FUSL method. A phase is called *significant* if the corresponding $\mathcal{G}_{FUSL}$ procedure terminates at Steps 2 or 3a), otherwise it is called *non-significant*. Clearly, if the value of $D_{v,y}$ is provided, which is the assumption made in Nesterov's smoothing scheme [5], then we can set $D_1 = D_{v,Y}$ in the schemes of both the USL and FUSL methods, and consequently, all the phases of both methods become significant.

As the same as the FAPL method, an iteration of procedure $\mathcal{G}_{FUSL}$ is also referred to an iteration of the FUSL method. The following result establishes a bound on the total number of iterations performed by the FUSL method to find an ϵ-solution to problem (2.1)–(2.31). Note that the proof of this result is similar to that of Theorem 7 in [1].

Theorem 2.8 *Suppose that $\{\alpha_k\}$ in procedure $\mathcal{G}_{FUSL}$ are chosen such that (2.10) holds. Then, the total number of iterations performed by the FUSL method for computing an ϵ-solution to problem (2.1)–(2.31) is bounded by*

$$\overline{N}(\epsilon) := S_1 + S_2 + \left(\frac{2}{\sqrt{2}-1} + \frac{\sqrt{2}}{1-q}\right)\frac{cR\|A\|}{\theta\beta\epsilon}\sqrt{\frac{\tilde{D}}{\sigma_v}} + \left(S_1 + \frac{1}{1-\sqrt{q}}\right)cR\sqrt{\frac{L_{\hat{f}}}{\theta\beta\epsilon}}, \tag{2.47}$$

where q and $D_{v,Y}$ are defined by (2.14) and (2.38) respectively, and

$$\tilde{D} := \max\{D_1, 2D_{v,Y}\},\ S_1 := \max\left\{\left\lceil \log_2 \frac{D_{v,Y}}{D_1}\right\rceil, 0\right\} \text{ and}$$
$$S_2 := \left\lceil \log_{\frac{1}{q}} \frac{4\sqrt{2}R\|A\|\sqrt{\frac{D_{v,Y}}{\sigma_v}} + 2R^2 L_{\hat{f}}}{\epsilon}\right\rceil. \tag{2.48}$$

Proof We prove this result by estimating the numbers of iterations performed within non-significant and significant phases separately. Suppose that the sets of indices of the non-significant and significant phases are $\{m_1, m_2, \ldots, m_{s_1}\}$ and $\{n_1, n_2, \ldots, n_{s_2}\}$ respectively. For any non-significant phase m_k, $1 \le k \le s_1$, we can easily see from Step 3b) that $D_{m_{k+1}} = 2D_{m_k}$, by part b) in Lemma 2.6, the number of non-significant phases performed by the FUSL method is bounded by S_1 defined above, i.e., $s_1 \le S_1$.

In addition, since $D_{m_{s_1}} \le \tilde{D}$, we have $D_{m_k} \le (1/2)^{s_1-k}\tilde{D}$, where $\tilde{D}$ is defined above. Combining the above estimates on s_1 and D_{m_k}, and in view of the fact $\Delta_{m_k} > \epsilon$ for all $1 \le k \le s_1$, we can bound the number of iterations performed within the non-significant phases by

$$\begin{aligned}\overline{N}_1 &= \sum_{k=1}^{s_1} \overline{N}(\Delta_{m_k}, D_{m_k}) \le \sum_{k=1}^{s_1} \overline{N}\left(\epsilon, \tilde{D}/2^{s_1-k}\right)\\ &\le S_1\left(cR\sqrt{\frac{L_{\hat{f}}}{\theta\beta\epsilon}} + 1\right) + \frac{\sqrt{2}cR\|A\|}{\theta\beta\epsilon}\sqrt{\frac{\tilde{D}}{\sigma_v}}\sum_{k=1}^{S_1} 2^{-\frac{(S_1-k)}{2}}\\ &\le S_1\left(cR\sqrt{\frac{L_{\hat{f}}}{\theta\beta\epsilon}} + 1\right) + \frac{2cR\|A\|}{(\sqrt{2}-1)\theta\beta\epsilon}\sqrt{\frac{\tilde{D}}{\sigma_v}}.\end{aligned} \tag{2.49}$$

Applying Lemma 8 in [1] and relation (2.32), and in view of the fact that $p_0, p_1 \in B(\overline{x}, R)$ in Algorithm 2, the initial gap is bounded by

$$\begin{aligned}\Delta_1 := \mathrm{ub}_1 - \mathrm{lb}_1 &\le \left[F(p_0) - F(p_1) - \langle F'(p_1), p_0 - p_1\rangle\right]\\ &\quad + \left[\hat{f}(p_0) - \hat{f}(p_1) - \left\langle \hat{f}'(p_1), p_0 - p_1\right\rangle\right] \qquad (2.50)\\ &\le 4\sqrt{2}R\|A\|\sqrt{\frac{D_{v,Y}}{\sigma_v}} + 2R^2 L_{\hat{f}}, \qquad (2.51)\end{aligned}$$

where $F'(p_1) \in \partial F(p_1)$. Then for the significant phases, similar to the proof of Theorem 2.5, we have $s_2 \le S_2$. Moreover, for any n_k, $1 \le k \le s_2$, using Lemmas 2.3, 2.6, we have $D_{n_k} \le \tilde{D}$, $\Delta_{n_{k+1}} \le q\Delta_{n_k}$, and $\Delta_{n_{s_2}} > \epsilon$, which implies $\Delta_{n_k} > \epsilon/q^{s_2-k}$. Combine these estimates on D_{n_k}, Δ_{n_k} and bound on s_2, we can see that the total number of iterations performed within the significant phases is bounded by

$$\begin{aligned}\overline{N}_2 &= \sum_{k=1}^{s_2} \overline{N}(\Delta_{n_k}, D_{n_k}) \le \sum_{k=1}^{s_2} \overline{N}(\epsilon/q^{s_2-k}, \tilde{D}) \\ &\le S_2 + cR\sqrt{\frac{L_{\hat{f}}}{\theta\beta\epsilon}} \sum_{k=1}^{S_2} q^{\frac{s_2-k}{2}} + \frac{\sqrt{2}cR\|A\|}{\theta\beta\epsilon}\sqrt{\frac{\tilde{D}}{\sigma_v}} \sum_{k=1}^{S_2} q^{s_2-k} \qquad (2.52) \\ &\le S_2 + \frac{cR}{1-\sqrt{q}}\sqrt{\frac{L_{\hat{f}}}{\theta\beta\epsilon}} + \frac{\sqrt{2}cR\|A\|}{\theta\beta\epsilon(1-q)}\sqrt{\frac{\tilde{D}}{\sigma_v}}.\end{aligned}$$

Finally, the total number of iterations performed by the FUSL method is bounded by $\overline{N}_1 + \overline{N}_2$, and thus (2.47) holds. □

From (2.47) in the above theorem, we can see that the iteration complexity of the FUSL method for solving problem (2.1)–(2.31) is bounded by

$$\mathcal{O}\left(\sqrt{\frac{L_{\hat{f}}}{\epsilon}} + \frac{\|A\|}{\epsilon}\right). \qquad (2.53)$$

Note that, similar to the FAPL method, the FUSL method requires two function evaluations for each iteration . The above iteration complexity is the same as that of the Nesterov smoothing scheme in [5] and the USL method in [1]. However, both the USL and FUSL methods improve Nesterov's smoothing scheme in that both of them are problem parameter free. In addition, as detailed in Subsection A below, the FUSL method further improves the USL method by reducing its iteration cost and improving the accuracy for solving its subproblems.

3 Solving unconstrained CP problems through ball-constrained CP problems

In this section we discuss the following unconstrained CP problem:

$$f^* := \min_{x\in\mathbb{R}^n} f(x), \qquad (3.1)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex, and for any closed sets $\Omega \in \mathbb{R}^n$, there exists $M(\Omega) > 0$ and $\rho(\Omega) \in [0, 1]$, such that

$$f(y) - f(x) - \langle f'(x), y - x\rangle \le \frac{M(\Omega)}{1+\rho(\Omega)}\|y - x\|^{1+\rho(\Omega)}, \quad \forall x, y \in \Omega. \qquad (3.2)$$

The above assumption on $f(\cdot)$ covers unconstrained nonsmooth ($\rho(\Omega) = 0$), smooth ($\rho(\Omega) = 1$) and weakly smooth ($0 < \rho(\Omega) < 1$) CP problems.

We first present a generic algorithmic framework to solve unconstrained CP problems through solutions to a series of ball-constrained CP problems in Sect. 3.1, then

extend the FAPL and FUSL methods to unconstrained CP problems with iteration complexity analysis in Sect. 3.2.

3.1 Expansion algorithm for unconstrained CP problems

In order to analyze the computational complexity of BL methods, it is commonly assumed that the feasible set of interest is compact (e.g., [1,20,22]). While the compactness assumption is essential for performing complexity analysis, the applicability of existing BL methods on unconstrained CP problem may be impaired. In practice, one possible workaround for BL methods to solve (3.1) may involve an assumption on the distance from a point $\bar{x}$ to an optimal solution x^*, namely, $x^* \in B(\bar{x}, R) := \{x \in \mathbb{R}^n : \|x - \bar{x}\| \leq R\}$ for some $\bar{x}$ and R. With such an assumption, we can solve (3.1) by considering a ball-constrained CP problem

$$f^*_{\bar{x},R} := \min_{x \in B(\bar{x},R)} f(x). \tag{3.3}$$

The above technique was also discussed in [32] (see Sect. 4 in [32]). It should be noted that, while the above equivalent reformulation seems straightforward, its computational complexity relies almost exclusively on the radius R. In particular, if R is close to the distance from $\bar{x}$ to X^*, the optimal solution set of (3.1), i.e., $R - D^*$ is small enough, where

$$x^* := \operatorname{argmin}_x \{\|\bar{x} - x\| : x \in X^*\} \quad \text{and} \quad D^* := \|\bar{x} - x^*\|, \tag{3.4}$$

then the computational complexity for computing approximate solutions to the ball-constrained CP problem (3.3) and the original unconstrained CP problem (3.1) are close, and it is definitely reasonable to solve (3.3) instead. However, if R is severely overestimated, the computational complexity for solving the ball-constrained CP (3.3) may become much higher than the optimal complexity bound that depends on D^*.

Based on the above discussion, we can conclude that a good BL method should tackle the unconstrained CP problem (3.1) from two perspectives. Firstly, without any satisfiable knowledge regarding D^*, such BL method should still be able to solve (3.1) with optimal complexity bound that depends on D^*. Secondly, if there exists an approximate distance R that is close to D^*, a good BL method should solve the ball-constrained problem (3.3) efficiently. We will consider a generic framework that follows the former perspective in this subsection, and then extend the BL method to solve (3.1) in the next subsection.

In this subsection, our goal is to design a generic algorithmic framework that follows the former perspective in the above discussion, and solve the unconstrained CP problem (3.1) through solutions to a series of ball-constrained CP problems. It should be noted that such concept indeed follows naturally from the two perspectives discussed above: if a BL method is computationally efficient for ball-constrained CP problems of form (3.3), starting from certain R for (3.3) and enlarging it by two each time, we will eventually reach a close enough estimate of D^* after logarithmic amount of times.

Given $\overline{x} \in \mathbb{R}^n$, $R > 0, \epsilon > 0$, let us assume that there exists a first-order algorithm, denoted by $\mathcal{A}(\overline{x}, R, \epsilon)$, which can find an ϵ-solution to (3.3). In other words, we assume that each call to $\mathcal{A}(\overline{x}, R, \epsilon)$ will compute a point $z \in B(\overline{x}, R)$ such that $f(z) - f^*_{\overline{x},R} \leq \epsilon$. Moreover, throughout this section, we assume that the number of (sub)gradient evaluations required by $\mathcal{A}(\overline{x}, R, \epsilon)$ for finding an ϵ-solution to (3.3) is bounded by

$$N_{\overline{x},R,\epsilon} := \frac{C_1(\overline{x}, R, f)R^{\alpha_1}}{\epsilon^{\beta_1}} + \frac{C_2(\overline{x}, R, f)R^{\alpha_2}}{\epsilon^{\beta_2}}, \tag{3.5}$$

where $\alpha_1 \geq \beta_1 > 0$ and $\alpha_2 \geq \beta_2 > 0$. $C_1(\overline{x}, R, f)$ and $C_2(\overline{x}, R, f)$ are constants that depend on f in (3.3) and nondecreasing with respect to R. For example, if f is a smooth convex function, ∇f is Lipschitz continuous in $\mathbb{R}^n$ with constant L, i.e., (3.2) holds with $\rho(\mathbb{R}^n) = 1$ and $M(\mathbb{R}^n) = L$, and we apply the APL method to (3.3), then we have only one term with $\alpha_1 = 1$, $\beta_1 = 1/2$, and $C_1(\overline{x}, R, f) = c\sqrt{L}$ in (3.5), where c is a universal constant. Observe that the two complexity terms in (3.5) will be useful for analyzing some structured CP problems in Sect. 2.2. It should also be noted that a more accurate estimate of $C_1(\overline{x}, R, f)$ is $cM(B(\overline{x}, R))$, since the Lipschitz constant $L = M(\mathbb{R}^n)$ throughout $\mathbb{R}^n$ is larger than or equal to the local Lipschitz constant on $B(\overline{x}, R)$.

Let $\mathcal{A}$ be the algorithm that can find an ϵ-solution to ball-constrained problem (3.3). By utilizing a novel guess and check procedure, we present an expansion algorithm for unconstrained convex optimizations as follows:

Algorithm 3 Expansion algorithm for unconstrained CP problems

Choose an arbitrary $r_1 > 1$ and compute the initial gap $\Delta_1 := f(\overline{x}) - \min_{x \in B(\overline{x}, r_1)} h(\overline{x}, x)$.
For $k = 1, 2, \ldots,$
Step 1. Solve $\overline{x}'_k = \mathcal{A}(\overline{x}, r_k, \Delta_k)$, then use $\bar{x}'_k$ as the initial point to solve $\overline{x}''_k = \mathcal{A}(\overline{x}, 2r_k, \Delta_k)$.
Step 2. If $f(\overline{x}'_k) - f(\overline{x}''_k) > \Delta_k$, update $r_k \leftarrow 2r_k$ and go to Step 1.
Step 3. Otherwise, output $\overline{x}^*_k = \overline{x}''_k$, and let $\Delta_{k+1} = \Delta_k/2$ and $r_{k+1} = r_k$.

Note that in order to reflect the flexibility we choose r_1 as any positive constant in Algorithm 3. However, from the theoretical complexity point of view, we prefer to starting with a smaller r_1 (i.e., a lower bound on D^*). In fact, given the target accuracy ϵ one can derive a theoretically viable selection of r_1 given as follows. For simplicity, assume that f has Lipschitz continuous gradient with constant L, we have $f(x_0) - f^* \leq \frac{L}{2}\|x_0 - x^*\|^2$. If $\|x_0 - x^*\| \leq \sqrt{2\epsilon/L}$, then x_0 already satisfies $f(x_0) - f^* \leq \epsilon$. Therefore, we can set $r_1 = \sqrt{2\epsilon/L}$. Such an estimate requires some prior information of L. It should be noted that an overestimate on L does not hurt the complexity bound, since r_1 is updated exponentially fast and will approach D^* quickly.

Steps 1 and 2 in Algorithm 3 constitute a loop for finding a pair of solutions $(\overline{x}'_k, \overline{x}''_k)$ satisfying $0 \leq f(\overline{x}'_k) - f(\overline{x}''_k) \leq \Delta_k$ for any $k \geq 1$. Since $\overline{x}'_k$ and $\overline{x}''_k$ are Δ_k-optimal solutions to $\min_{x \in B(\overline{x}, r_k)} f(x)$ and $\min_{x \in B(\overline{x}, 2r_k)} f(x)$ respectively, this loop must terminate in finite time, because it will terminate whenever $r_k \geq D^*$.

For simplicity, we call it an expansion if we double the radius in Step 2. Each iteration may contain several expansions before outputting solution $\overline{x}_k^*$ in Step 3. Note that, for the implementation of Algorithm 3, most computational work exists in Step 1, which involves a sequence of calls to algorithm $\mathcal{A}$. However, notice that the gaps (Δ_k) and radiuses (r_k) for these calls are monotonically decreasing and increasing, respectively, numerous computational cost could be saved by using results from previous expansions and iterations. For instance, the output solutions of previous calls to $\mathcal{A}$ associated with larger gaps or smaller radiuses could always be used as the starting point for the current call to $\mathcal{A}$. For successive expansions, if the previous execution of Step 1 called $\mathcal{A}(\bar{x}, r_k, \Delta_k)$ and $\mathcal{A}(\bar{x}, 2r_k, \Delta_k)$ and the current execution of Step 1 calls $\mathcal{A}(\bar{x}, 2r_k, \Delta_k)$ and $\mathcal{A}(\bar{x}, 4r_k, \Delta_k)$, the computation of call $\mathcal{A}(\bar{x}, 2r_k, \Delta_k)$ could be saved. Moreover, if $\mathcal{A}$ is referred to some specific algorithms, such as BL type methods, more previous results, like the lower bounds and prox-centers, could also be utilized to save computational cost.

Note that there are proximal bundle methods incorporating with trust region techniques for updating the new iterate [35,36]. In [35] an quadratic cutting plane model based method and in [36] the idea of Chebychev center were used to generate the trust regions. These trust region methods restrict the iterates in the trust region for better convergence to the optimal solution, while the approximate solutions in the searching balls generated by our expansion algorithm are used only for checking whether or not the current searching ball needs to be expanded in order to get a better estimate of D^*.

Before analyzing the iteration complexity of Algorithm 3, we discuss some important observations related to the aforementioned expansions.

Lemma 3.1 *Let $\bar{x} \in \mathbb{R}^n$ and $r > 0$ be a constant, $\overline{x}_1$ and $\overline{x}_2$ be Δ-solutions to CP problems*

$$f^*_{\bar{x},r} := \min_{x \in B(\bar{x},r)} f(x) \quad \text{and} \quad f^*_{\bar{x},2r} := \min_{x \in B(\bar{x},2r)} f(x), \tag{3.6}$$

respectively. If $0 \le f(\overline{x}_1) - f(\overline{x}_2) \le \Delta$, then we have

$$f(\overline{x}_2) - f^* \le \left(3 + \frac{2D^*}{r}\right)\Delta, \tag{3.7}$$

where f^* and D^* are defined in (3.1) and (3.4) respectively.

Proof Clearly, by definition, we have $\|\overline{x}_1 - \bar{x}\| \le r, \|\overline{x}_2 - \bar{x}\| \le 2r, 0 \le f(\overline{x}_1) - f^*_{\bar{x},r} \le \Delta$ and $0 \le f(\overline{x}_2) - f^*_{\bar{x},2r} \le \Delta$. It suffices to consider the case when $f^*_{\bar{x},2r} > f^*$ and $\|x^* - \bar{x}\| > 2r$, since otherwise (3.7) holds trivially. Suppose x_1^* and x_2^* are the solutions to the first and second problems in (3.6) respectively, let $\hat{x}$ be the intersection of the line segment (x^*, x_1^*) with the ball $B(\bar{x}, 2r)$, and denote $R_1 := \|\hat{x} - x_1^*\|$ and $R_2 := \|x^* - x_1^*\|$. Clearly, $\hat{x} = (1 - \frac{R_1}{R_2})x_1^* + \frac{R_1}{R_2}x^*$. By the convexity of $f(\cdot)$, we have

$$f(\hat{x}) \le \left(1 - \frac{R_1}{R_2}\right) f(x_1^*) + \frac{R_1}{R_2} f(x^*), \tag{3.8}$$

which implies that

$$\frac{R_1}{R_2}\left[f(x_1^*) - f(x^*)\right] \le f(x_1^*) - f(\hat{x}), \tag{3.9}$$

and that $f(\hat{x}) \le f(x_1^*)$ due to the fact that $f(x^*) \le f(x_1^*)$. Also, we have $f(\hat{x}) \ge f(x_2^*)$ since $\hat{x} \in B(\overline{x}, 2r)$. In addition,

$$f(x_1^*) - f(x_2^*) = [f(x_1^*) - f(\overline{x}_1)] + [f(\overline{x}_1) - f(\overline{x}_2)] + [f(\overline{x}_2) - f(x_2^*)] \tag{3.10}$$

$$\le 0 + \Delta + \Delta = 2\Delta. \tag{3.11}$$

Combining the previous inequalities, we obtain

$$f(x_1^*) - 2\Delta \le f(x_2^*) \le f(\hat{x}) \le f(x_1^*), \tag{3.12}$$

which implies that $f(x_1^*) - f(\hat{x}) \le 2\Delta$. Using (3.9), and the fact that $R_1 \ge r$ and $R_2 \le D^* + r$, we have

$$f(x_1^*) - f(x^*) \le \frac{2R_2\Delta}{R_1} \le \left(2 + \frac{2D^*}{r}\right)\Delta.$$

Therefore,

$$f(\overline{x}_2) - f(x^*) \le f(\overline{x}_1) - f(x^*) \le [f(\overline{x}_1) - f(x_1^*)] + [f(x_1^*) - f(x^*)]$$
$$\le \left(3 + \frac{2D^*}{r}\right)\Delta. \qquad \square$$

We are now ready to prove the iteration complexity of Algorithm 3 for solving the unconstrained CP problem (3.1).

Theorem 3.2 *Suppose that the number of (sub)gradient evaluations required by $\mathcal{A}(\overline{x}, R, \epsilon)$ for finding an ϵ-solution to (3.3) is bounded by (3.5). For any $k \ge 1$, denote $\epsilon_k := f(\overline{x}_k^*) - f^*$ for the output $\overline{x}_k^*$ in Algorithm 3. Then we have*

(a) $r_k \le \bar{r} := \max\{r_1, 2D^*\}$ *for all $k \ge 1$, where D^* is defined in (3.4);*
(b) $\lim_{k\to\infty} \epsilon_k = 0$;
(c) *The total number of (sub)gradient evaluations performed by Algorithm 3 for finding the ϵ_k-solution $\bar{x}_k^*$ to problem (3.1) is bounded by*

$$\mathcal{O}\left(\frac{C_1(\overline{x}, 2\bar{r}, f)\bar{r}^{\alpha_1}}{\epsilon_k^{\beta_1}} + \frac{C_2(\overline{x}, 2\bar{r}, f)\bar{r}^{\alpha_2}}{\epsilon_k^{\beta_2}}\right). \tag{3.13}$$

Proof We start by proving a), if $r_1 \ge D^*$, then $f(\bar{x}_k') - f(\bar{x}_k'') \le \Delta_k$ for any $k \ge 1$ and no expansion takes place, hence $r_k = r_1 = \bar{r}$. If $r_1 \le D^*$, from Algorithm 3, we see that expansion occurs at Step 2 if and only if $f(\bar{x}_k') - f(\bar{x}_k'') > \Delta_k$. Hence,

if $r_k \geq D^*$, this condition is not satisfied and no more expansion is performed. This implies $r_k < 2D^*$.

To prove (b), observe that $\bar{x}'_k$ is used as the initial point for computing $\bar{x}''_k$ in Algorithm 3 and hence $f(\bar{x}''_k) \leq f(\bar{x}'_k)$. Combining this observation with the condition in Step 3, we have $0 \leq f(\bar{x}'_k) - f(\bar{x}''_k) \leq \Delta_k$. Applying Lemma 3.1 implies

$$\epsilon_k = f(\bar{x}''_k) - f^* \leq \left(3 + \frac{2D^*}{r_k}\right)\Delta_k. \tag{3.14}$$

Note that the total number of expansion is bounded by

$$\overline{S}_1 := \left\lceil \log_2 \frac{\bar{r}}{r_1} \right\rceil, \tag{3.15}$$

hence Δ_k decreases to 0 as k increases, we have $\lim_{k\to\infty} \epsilon_k = 0$.

To prove c), assume that the number of executions of Step 1 in Algorithm 3 for finding $\bar{x}^*_k$ is K. For any $1 \leq j \leq K$, let $\mathcal{A}(\bar{x}, R_j, \delta_j)$ and $\mathcal{A}(\bar{x}, 2R_j, \delta_j)$ be called in the j^{th} execution of Step 1. By using (3.5) and noting that $C_1(\bar{x}, R, f)$ and $C_2(\bar{x}, R, f)$ are nondecreasing with respect to R, we have the number of (sub)gradient evaluations performed by the j^{th} execution of Step 1 is bounded by

$$N_j := \frac{(1+2^{\alpha_1})C_1(\bar{x}, 2R_j, f)R_j^{\alpha_1}}{\delta_j^{\beta_1}} + \frac{(1+2^{\alpha_2})C_2(\bar{x}, 2R_j, f)R_j^{\alpha_2}}{\delta_j^{\beta_2}}. \tag{3.16}$$

Let N'_j and N''_j be the first and second terms on the right of (3.16), respectively. The $(j+1)^{th}$ execution of Step 1 either doubles the radius or reduces the gap by half comparing to the j^{th} execution, i.e., $R_{j+1} = 2R_j$ or $\delta_{j+1} = \delta_j/2$ respectively. Therefore we have either $N'_{j+1} \geq 2^{\alpha_1} N'_j$ and $N''_{j+1} \geq 2^{\alpha_2} N''_j$, or $N'_{j+1} \geq 2^{\beta_1} N'_j$ and $N''_{j+1} \geq 2^{\beta_2} N''_j$. Since $2^{\alpha_1} \geq 2^{\beta_1} > 1$ and $2^{\alpha_2} \geq 2^{\beta_2} > 1$, we can combine these two cases and have

$$N'_j \leq 2^{-\beta_1} N'_{j+1} \text{ and } N''_j \leq 2^{-\beta_2} N''_{j+1}, \text{ for } 1 \leq j \leq K-1, \tag{3.17}$$

which further implies

$$N'_j \leq 2^{-\beta_1(K-j)} N'_K \text{ and } N''_j \leq 2^{-\beta_2(K-j)} N''_K, \text{ for } 1 \leq j \leq K. \tag{3.18}$$

Then the total number of (sub)gradient evaluations performed by these K executions of Step 1 in Algorithm 3 is bounded by

$$N := \sum_{j=1}^{K}(N'_j + N''_j) \le N'_K \sum_{j=1}^{K} 2^{-\beta_1(K-j)} + N''_K \sum_{j=1}^{K} 2^{-\beta_2(K-j)} \tag{3.19}$$

$$< N'_K \sum_{j=0}^{+\infty} 2^{-\beta_1 j} + N''_K \sum_{j=0}^{+\infty} 2^{-\beta_2 j} \le \frac{1}{1-2^{-\beta_1}} N'_K + \frac{1}{1-2^{-\beta_2}} N''_K \tag{3.20}$$

$$\le \frac{(1+2^{\alpha_1})C_1(\overline{x}, 2r_k, f)}{1-2^{-\beta_1}} \cdot \frac{r_k^{\alpha_1}}{\Delta_k^{\beta_1}} + \frac{(1+2^{\alpha_2})C_2(\overline{x}, 2r_k, f)}{1-2^{-\beta_2}} \cdot \frac{r_k^{\alpha_2}}{\Delta_k^{\beta_2}}. \tag{3.21}$$

The last inequality follows from the observation that the last (i.e., K^{th}) execution of Step 1 before $\bar{x}_k^*$ is found has $R_K = r_k$ and $\delta_K = \Delta_k$. Combining the above inequality with (3.14), we have

$$N < \sum_{i=1}^{2} \frac{(1+2^{\alpha_i})C_i(\overline{x}, 2r_k, f)}{1-2^{-\beta_i}} \cdot \frac{r_k^{\alpha_i}\left(3+\frac{2D^*}{r_k}\right)^{\beta_i}}{\epsilon_k^{\beta_i}}. \tag{3.22}$$

Since $\alpha_i \ge \beta_i > 0, C_i(\overline{x}, 2r_k, f) r_k^{\alpha_i}(3+\frac{2D^*}{r_k})^{\beta_i} = C_i(\overline{x}, 2r_k, f) r_k^{\alpha_i-\beta_i}(3r_k+2D^*)^{\beta_i}$ for $i = 1, 2$ are monotonically increasing with respect to r_k, which, in view of the fact $r_k < 2\bar{r}$ proved by part a), therefore clearly implies

$$N < \sum_{i=1}^{2} \frac{(2^{3\beta_i} + 2^{\alpha_i+3\beta_i})C_i(\overline{x}, 2\bar{r}, f)}{2^{\beta_i}-1} \cdot \frac{\bar{r}^{\alpha_i}}{\epsilon_k^{\beta_i}}. \tag{3.23}$$

Hence the proof is complete. □

Note that to solve the unconstrained CP problem (3.1), the termination criterions of most first-order algorithms are based on the residual of the (sub)gradient, which would lead to different complexity analysis. To the best of our knowledge, without any prior information on D^*, there is no verifiable termination criterion based on functional optimality gap that could guarantee the termination of algorithms for finding an ϵ-solution to (3.1). Comparing to Nesterov's optimal gradient method for unconstrained problems in [6], Algorithm 3 only provides efficiency estimates about $\epsilon_k := f(\overline{x}_k^*) - f^*$ when the output $\overline{x}_k^*$ is updated, while the optimal gradient method could have estimates about $\overline{\epsilon}_k := f(x_k) - f^*$ for each iterate x_k. For both methods the efficiency estimates involve D^*. Since Algorithm 3 extends methods for ball-constrained CP problems to solve (3.1), and the iterations in the expansions of Algorithm 3 could be regarded as a guess and check procedure to estimate D^*, it is reasonable that the efficiency estimates are only provided for unexpansive steps, i.e., Step 3 of Algorithm 3, which output $\overline{x}_k^*$.

It has been shown in [1] that the APL method and its variant, the USL method, achieve the optimal iteration complexities in (3.3) for smooth, nonsmooth and weakly smooth CP problems and a class of structured saddle point problems respectively, on any convex and compact feasible set. So Algorithm 3 could be incorporated to solve

(3.1) with the optimal iteration complexities too. Therefore, the remaining part of this paper will focus on how to improve the efficiency of these BL type methods for solving ball-constrained CP problems.

3.2 Extending FAPL and FUSL for unconstrained problems

In this subsection, we study how to utilize the FAPL and FUSL methods to solve the unconstrained problems based on our results in Sect. 3.

Let us first consider the case when f in (3.1) satisfies (3.2). If the method $\mathcal{A}$ in Step 1 of Algorithm 3 is given as the FAPL method, then by Theorem 2.5, and the fact that only one (sub)gradient of f is computed in each iteration of the FAPL method, the number of (sub)gradient evaluations within one call to $\mathcal{A}(\overline{x}, R, \epsilon)$ is bounded by $N(\epsilon)$ given by (2.33).

Therefore, for the FAPL method, we have $\alpha_1 = \frac{2(1+\rho)}{3+2\rho}$, $\beta_1 = \frac{2}{1+3\rho}$, $C_1(\overline{x}, R, f) = C' M^{\frac{2}{1+3\rho}}$ and $C_2(\overline{x}, R, f) = \alpha_2 = \beta_2 = 0$ in (3.5), where C' is a constant depending on the parameters q, θ, β, ρ and c in the FAPL method. Letting $\epsilon_k := f(\overline{x}_k^*) - f^*$ for $k \geq 1$ in Algorithm 3 and applying Theorem 3.2, we then conclude for finding the ϵ_k-solution $\overline{x}_k^*$ to problem (3.1), the total number of (sub)gradient evaluations performed by Algorithm 3 is bounded by

$$\mathcal{O}\left(\left[\frac{M(2\overline{r})^{1+\rho}}{\epsilon_k}\right]^{\frac{2}{1+3\rho}}\right), \tag{3.24}$$

where $M := M(B(\overline{x}, 2\overline{r}))$, $\rho := \rho(B(\overline{x}, 2\overline{r}))$ and $\overline{r}$ is defined in part a) of Theorem 3.2. It should be noted that the constants M and ρ are local constants that depend on the size of the initial ball, i.e., r_1, and the distance from $\overline{x}$ and x^*, which are not required for the FAPL method and Algorithm 3, and also generally smaller than the constants $M(\mathbb{R}^n)$ and $\rho(\mathbb{R}^n)$, respectively, for the global Hölder continuity condition.

Moreover, if f in (3.1) is given in the form of (2.31) as a structured nonsmooth CP problem, then the FUSL method could be applied to solve the corresponding structured ball-constrained problems in Algorithm 3. By Theorem 2.8, the number of (sub)gradient evaluations of f within one call to $\mathcal{A}(\overline{x}, R, \epsilon)$ is bounded by

$$S_1 + S_2 + C' R \sqrt{\frac{L_{\hat{f}}}{\epsilon}} + \frac{C'' R \|A\|}{\epsilon}, \tag{3.25}$$

where C', C'' are some constants depending on the parameters $q, \theta, \beta, \sigma_v, c, D_0$ and $D_{v,Y}$ in the FUSL method.

Applying Theorem 3.2 with $\alpha_1 = \alpha_2 = 1, \beta_1 = \frac{1}{2}, \beta_2 = 1, C_1(\overline{x}, R, f) = C'\sqrt{L_{\hat{f}}}$, and $C_2(\overline{x}, R, f) = C''\|A\|$, we conclude that the total number of (sub)gradient evaluations performed by Algorithm 3 to find the ϵ_k-solution $\overline{x}_k^*$ to problem (3.1)-(2.31) is bounded by

$$\mathcal{O}\left(2C'\overline{r}\sqrt{\frac{L_{\hat{f}}}{\epsilon_k}}+\frac{2C''\overline{r}\|A\|}{\epsilon_k}\right). \tag{3.26}$$

Similar to the FAPL method, here $L_f := L_f(B(\overline{x}, 2\overline{r}))$ is a lower bound of $L_f(\mathbb{R}^n)$.

4 Numerical experiments

In this section we apply the FAPL and FUSL methods to solve a few large-scale CP problems, including the quadratic programming problems with large Lipschitz constants, synthetic and real world total variation based image reconstruction problems, then compare them with some other first-order algorithms. All the algorithms were implemented in MATLAB, Version R2011a and all experiments were performed on a desktop with an Inter Dual Core 2 Duo 3.3 GHz CPU and 8G memory. In Sects. 4.1 and 4.2, some random matrices are generated during the numerical experiments.

4.1 Quadratic programming

The main purpose of this section is to investigate the performance of the FAPL method for solving smooth CP problems especially with large Lipschitz constants and demonstrate the improvement of the FAPL method comparing to some other BL type methods. And since most BL type methods require compact feasible sets, we consider the following quadratic programming problem:

$$\min_{\|x\|\leq 1} \|Ax-b\|^2, \tag{4.1}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. We compare the FAPL method with NERML [22] and APL [1]. We also compare the FAPL method with the built-in Matlab linear system solver . In the APL method, the subproblems are solved by MOSEK [37], an efficient software package for linear and second-order cone programming. Two cases with different choices of initial lower bound LB in these experiments are conducted: (1). $LB = 0$ and (2). $LB = -\infty$. Moreover, for all the instances, except for the worst-case instance where the given data is specially designed and has no randomness, we randomly generate the data and run the algorithms 100 times, then the means and standard deviations of the number of iterations, CPU time and accuracy of solution are computed and reported for each algorithm.

In our experiments, given m and n, two types of matrix A are generated. The first type of matrix A is randomly generated with entries uniformly distributed in [0,1], while the entries of the second type are normally distributed according to $N(0, 1)$. We then randomly choose an optimal solution x^* within the unit ball in $\mathbb{R}^n$, and generate the data b by $b = Ax^*$. We apply FAPL, NERML and APL to solve (4.1) with the set of data A and b, and the accuracy of the solutions are measured by $e_k = \|Ax_k - b\|^2$. Additionally, for each type of matrix A, we run one instance 30 times and compute the means and standard deviations of the numbers of iterations, CPU time and accuracies. The results are shown in Tables 2 and 3.

Table 2 Uniformly distributed QP instances

Alg	LB	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)
$A: n = 4000, m = 3000, L \approx 2.0e6, e_0 \approx 3.0e4$							
FAPL	0	(102.5, 3.3)	(4.2, 0.5)	(8.8e−7, 7.6e−8)	(141.1, 4.0)	(5.6, 0.6)	(8.7e−9, 8.6e−10)
	$-\infty$	(254.3, 40.3)	(8.1, 1.4)	(8.1e−7, 1.4e−7)	(435.8, 54.2)	(13.8, 1.9)	(8.1e−9, 1.6e−9)
APL	0	(128.2, 4.1)	(33.6, 2.0)	(9.1e−7, 5.8e−8)	(174.8, 5.0)	(46.3, 2.8)	(9.3e−9, 5.5e−10)
	$-\infty$	(408.6, 35.7)	(103.4, 10.7)	(6.8e−7, 2.1e−7)	(717.1, 52.0)	(187.5, 16.4)	(7.0e−9, 1.9e−9)
NERML	0	(219.8, 5.2)	(51.1, 3.0)	(9.2e−7, 6.4e−8)	(287.7, 7.0)	(67.8, 4.2)	(9.3e−9, 6.0e−10)
	$-\infty$	(1000, 0)	(257.1, 16.5)	(1.2e−3, 5.3e−4)	(2000, 0)	(526.6, 35.4)	(8.1e−4, 3.7e−4)
$A: n = 8000, m = 4000, L \approx 8.0e6, e_0 \approx 7.0e4$							
FAPL	0	(63.9, 2.9)	(6.8, 0.7)	(8.2e−7, 1.1e−7)	(80.8, 3.3)	(8.5, 0.8)	(8.3e−9, 1.1e−9)
	$-\infty$	(153.7, 22.1)	(13.0, 2.1)	(7.7e−7, 1.6e−7)	(228.7, 28.6)	(19.3, 2.7)	(7.6e−9, 1.8e−9)
APL	0	(78.0, 2.1)	(60.5, 2.8)	(8.9e−7, 6.9e−8)	(99.6, 2.6)	(78.3, 3.7)	(8.8e−9, 8.0e−10)
	$-\infty$	(259.5, 17.9)	(187.4, 14.8)	(6.4e−7, 2.4e−7)	(420.0, 37.1)	(318.7, 31.8)	(7.1e−9, 2.2e−9)
NERML	0	(155.3, 2.8)	(108.4, 4.2)	(8.7e−7, 7.9e−8)	(189.7, 3.2)	(133.6, 5.2)	(8.9e−9, 8.2e−10)
	$-\infty$	(1000, 0)	(782.9, 44.3)	(1.9e−4, 1.8e−4)	(2000, 0)	(1594.2, 91.6)	(1.3e−4, 1.1e−4)
$A \in \mathbb{R}^{m\times n}$ m=10000	LB	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)
FAPL method for large dimension matrix							
$n = 20000$	0	(100.0, 2.2)	(173.7, 7.7)	(8.2e−11, 1.1e−11)	(142.4, 2.6)	(247.0, 9.4)	(8.4e−16, 1.1-16)
$L \approx 5.0e7$	$-\infty$	(201.5, 23.4)	(335.7, 41.8)	(7.8e−8, 1.7e−8)	(557.8, 44.9)	(934.7, 81.9)	(7.1e−15, 2.0e−15)
$n = 40000$	0	(69.2, 1.8)	(244.3, 14.9)	(7.9e−11, 1.4e−11)	(97.5, 2.6)	(342.8, 19.8)	(7.8e−16, 1.3e−16)
$L \approx 1.0e8$	$-\infty$	(131.9, 12.3)	(438.8, 45.2)	(7.1e−8, 2.2e−8)	(330.9, 29.4)	(1111.1, 110.8)	(6.6e−15, 2.3e−15)
$n = 60000$	0	(58.1,0.7)	(309.3, 18.6)	(9.5e−11, 3.3e−12)	(78.1, 0.6)	(414.1, 22.5)	(7.5e−16, 2.0e−17)
$L \approx 1.5e8$	$-\infty$	(119.8, 13.6)	(595.8, 71.4)	(6.3e−8,2.5e−8)	(279.5,25.4)	(1411.0, 138.6)	(7.1e−15, 2.1e−15)

In order to investigate the efficiency of solving unconstrained CP problems using the proposed expansion algorithm (i.e., Algorithm 3), we conduct two sets of experiments to compare the performance of solving the unconstrained QP problem $\min_{x\in\mathbb{R}^n} \|Ax-b\|^2$ using two different strategies: one starts with small initial feasible ball, then applies the unconstrained FAPL method (i.e., incorporating the expansion algorithm with the FAPL method as subroutine $\mathcal{A}$), while the other one, under the assumption that a bound on D^* defined in (3.4) is known, applies the ball-constrained FAPL method directly by choosing some large ball that contains at least one optimal solution.

Since the performance of both methods would be affected by the distance between the initial point x_0 and the optimal solution set, in both experiments, we set $\bar{x} = 0$, $D^* \approx 1$, and for any initial ball $B(\bar{x}, R)$, we choose the starting point x_0 randomly within the ball and then normalize x_0 and set $\|x_0\| = \frac{R}{2}$. For both methods, the initial

Table 3 Gaussian distributed QP instances

Alg	LB	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)
$A: n = 4000, m = 3000, L \approx 2.3e4, e_0 \approx 2.0e3$							
FAPL	0	(101.7, 2.2)	(4.2, 0.4)	(8.9e−7, 7.0e−8)	(134.1, 2.9)	(5.5, 0.5)	(8.8e−9, 8.2e−10)
	$-\infty$	(291.7, 37.5)	(9.3, 1.4)	(7.8e−7, 1.6e−7)	(481.1, 55.8)	(15.3, 2.1)	(7.8e−9, 1.5e−9)
APL	0	(124.4, 3.4)	(33.3, 1.9)	(9.1e−7, 6.0e−8)	(164.0, 4.2)	(44.3, 2.6)	(9.2e−9, 5.9e−10)
	$-\infty$	(593.6, 57.6)	(158.0, 16.4)	(6.5e−7, 2.2e−7)	(937.2, 99.9)	(254.4, 31.5)	(7.1e−9, 2.2e−9)
NERML	0	(193.5, 5.5)	(46.1, 2.5)	(9.1e−7, 6.3e−8)	(252.8, 6.4)	(60.7, 3.3)	(9.0e−9, 7.2e−10)
	$-\infty$	(1000, 0)	(262.7, 15.8)	(3.3e−2, 1.8e−2)	(2000, 0)	(537.2, 34.0)	(2.3e−2, 1.3e−2)
$A: n = 8000, m = 4000, L \approx 2.3e4, e_0 \approx 2.0e3$							
FAPL	0	(48.7, 1.1)	(5.2, 0.5)	(8.2e−7, 1.1e−7)	(61.5, 1.2)	(6.5, 0.6)	(8.4e−9, 1.1e−9)
	$-\infty$	(151.5, 18.3)	(12.7, 1.6)	(7.4e−7, 1.7e−7)	(221.2, 20.9)	(18.5, 1.9)	(6.9e−9, 2.1e−9)
APL	0	(56.7, 2.1)	(44.4, 2.3)	(8.5e−7, 9.1e−8)	(71.9, 2.2)	(56.9, 2.8)	(8.5e−9, 8.9e−10)
	$-\infty$	(356.9, 24.0)	(279.2, 22.3)	(6.1e−7, 2.3e−7)	(536.9, 58.9)	(427.9, 55.0)	(6.9e−9, 2.4e−9)
NERML	0	(107.5, 2.4)	(75.5, 3.2)	(8.6e−7, 7.8e−8)	(134.8, 2.7)	(95.5, 3.9)	(8.7e−9, 8.5e−10)
	$-\infty$	(1000, 0)	(796.7, 41.2)	(4.2e−3, 3.7e−3)	(2000, 0)	(1611.5, 85.3)	(3.3e−2, 3.1e−3)
$A \in \mathbb{R}^{m\times n}$ m=10000	LB	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)
FAPL method for large dimension matrix							
$n = 20000$	0	(77.4, 1.1)	(135.5, 6.8)	(8.1e−11, 1.1e−11)	(110.1, 1.3)	(192.1, 8.8)	(8.1e−16, 1.1-16)
$L \approx 6.0e4$	$-\infty$	(179.1, 8.5)	(298.1, 16.9)	(7.3e−8, 2.0e−8)	(525.6, 41.2)	(882.8, 71.1)	(7.3e−15, 2.0e−15)
$n = 40000$	0	(48.0, 0.0)	(169.2, 10.1)	(6.6e−11, 4.5e−12)	(67.2, 0.4)	(235.5, 11.7)	(8.6e−16, 1.3e−16)
$L \approx 9.0e4$	$-\infty$	(114.1, 10.6)	(380.2, 37.0)	(6.5e−8, 2.4e−8)	(282.6, 32.0)	(947.3, 121.9)	(6.5e−15, 2.3e−15)
$n = 60000$	0	(34.2, 0.4)	(186.4, 17.4)	(7.9e−11, 1.7e−11)	(49.8, 0.4)	(269.0, 21.8)	(6.4e−16, 1.5e−16)
$L \approx 1.2e5$	$-\infty$	(94.2, 5.5)	(471.1, 33.2)	(6.0e−8,2.4e−8)	(232.0,22.5)	(1176.8, 119.8)	(5.9e−15, 2.6e−15)

lower bound is set to $-\infty$, and the parameters of the FAPL method are the same. In this first experiment, A is generated randomly with entries uniformly distributed in [0, 1]. In the second experiment, we use the worst-case QP instance for first-order methods which are generated by A. Nemirovski (see the construction scheme in [7] and [4]). When we apply the unconstrained FAPL method, the radiuses of the initial balls are chosen as $10^{-5}D$, $10^{-4}D$, $10^{-3}D$, $10^{-2}D$ and $10^{-1}D$, respectively. While the ball-constrained FAPL method is employed, the radiuses of the balls are selected as $10^5 D$, $10^4 D$, $10^3 D$, $10^2 D$ and $10D$, respectively. The results are shown in Table 5.

The advantages of the FAPL method can be observed from these experiments. Firstly, among these three BL type methods, NERML requires more iterations than APL and FAPL, which have optimal iteration complexity for this problem. Moreover, in view of the CPU time, FAPL has 10 times cheaper computational cost per iteration than that of APL and NERML.

Table 4 Comparison to Matlab solver

Matrix A:$m \times n$	Matlab $A\backslash b$		FAPL method		
	Time (mean, std)	$\|Ax-b\|$ (mean, std)	Iter. (mean, std)	Time (mean, std)	$\|Ax-b\|$ (mean, std)
Uniform 2000×4000	(3.6, 0.2)	(2.1e−12, 5.9e−13)	(196.7, 3.4)	(5.5, 0.4)	(1.3e−11, 7.3e−13)
Uniform 2000×6000	(6.5, 0.3)	(2.6e−12, 6.9e−13)	(155.2, 4.5)	(6.1, 0.5)	(1.3e−11, 8.3e−13)
Uniform 2000×8000	(9.5, 0.3)	(3.1e−12, 7.1e−13)	(135.2, 4.3)	(6.8, 0.6)	(1.3e−11, 8.7e−13)
Uniform 2000×10000	(12.5, 0.4)	(3.6e−12, 8.3e−13)	(118.2, 8.8)	(7.3, 0.8)	(1.3e−11, 8.5e−13)
Gaussian 2000×4000	(3.6, 0.2)	(4.1e−13, 5.5e−14)	(151.3, 1.8)	(4.2, 0.3)	(1.3e−11, 8.1e−13)
Gaussian 2000×6000	(6.5, 0.5)	(4.5e−13, 5.7e−14)	(94.8, 0.5)	(3.8, 0.3)	(1.2e−11, 1.1e−12)
Gaussian 2000×8000	(9.5, 0.5)	(4.6e−13, 5.5e−14)	(92.0, 0.2)	(4.7, 0.4)	(1.2e−11, 8.3e−13)
Gaussian 2000×10000	(12.6, 0.6)	(4.6e−13, 5.8e−14)	(83.6, 1.1)	(5.3, 0.5)	(1.2e−11, 1.1e−12)

Secondly, consider the difference between the performance of setting the initial lower bound equal to 0 and $-\infty$, it is also evident that FAPL is more robust to the choice of the initial lower bound and it updates the lower bound more efficiently than the other two BL type methods. Though setting the initial lower bound equal to $-\infty$ increases the numbers of iterations for all these three BL type methods, a close examination reveals that the difference between setting the lower bound to zero and $-\infty$ for FAPL is not so significant as that for APL and NERML, especially for large matrix, for example, the second one in Table 2 .

Thirdly, FAPL needs less number of iterations than APL, especially when the required accuracy is high. A plausible explanation is that exactly solving the subproblems provides better updating for the prox-centers, and consequently, more accurate prox-centers improve the efficiency of algorithm . The experiments show that, for APL and NERML, it is hard to improve the accuracy beyond 10^{-10}. However, FAPL can keep almost the same speed for deceasing the objective value from 10^6 to 10^{-21}.

Fourthly, we can clearly see from Table 4 that FAPL is comparable to or outperforms the built-in Matlab solver for randomly generated linear systems, even though our code is implemented in MATLAB rather than lower-level languages, such as C or FORTRAN. We can expect that the efficiency of FAPL will be improved by using C or FORTRAN implementation, which has been used in the MATLAB solver for linear systems.

Finally, from Table 5 it is evident that the performance of both the unconstrained FAPL method and the ball-constrained FAPL method are affected by the distance between the starting point x_0 and the optimal solution set. And improper estimations on D^* would increase the computational cost significantly. Comparing the results presented in the same rows of the left and right columns in Table 5, one can see that, for the uniform instance, both methods could achieve high accuracy of solution, and the ball-constrained FAPL method needs less iterations and CPU time than the unconstrained FAPL method, but for the worst-case instance, the unconstrained FAPL method outperforms the the ball-constrained FAPL method more significant in terms of accuracy of solution, iterations and CPU time.

Table 5 Unconstrained QP instances

Uniform instance: $A = rand(4000, 8000)$

FAPL-Unconstrained				FAPL-Ball constrained			
Radius	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)	Radius	Iter. (mean, std)	Time (mean, std)	Acc. (mean, std)
1e−5	(1377.0, 55.4)	(110.2, 6.8)	(7.1e−11, 2.0e−11)	1e5	(966.1, 69.3)	(79.4, 5.9)	(7.2e−11, 1.8e−11)
1e−4	(1114.8, 88.2)	(88.7, 8.0)	(7.1e−11, 2.0e−11)	1e4	(947.6, 74.0)	(78.2, 6.6)	(7.1e−11, 1.8e−11)
1e−3	(918.4, 73.5)	(73.5, 6.8)	(7.3e−11, 2.0e−11)	1e3	(828.9, 71.0)	(68.4, 6.6)	(7.1e−11, 2.0e−11)
1e−2	(900.9, 73.6)	(72.8, 6.6)	(7.3e−11, 2.0e−11)	1e2	(667.2, 54.4)	(54.7, 4.9)	(7.2e−11, 1.9e−11)
1e−1	(834.3, 66,7)	(67.7, 6.1)	(7.3e−11, 2.0e−11)	1e1	(531.9, 41.9)	(43.4, 3.9)	(7.3e−11, 1.7e−11)

Worst-case instance: $A = Bdata(2062, 4124)$

FAPL-Unconstrained				FAPL-Ball constrained			
Radius	Iter.	Time	Acc.	Radius	Iter.	Time	Acc.
1e−5	1385	27.9	9.93e−8	1e5	4000	76.2	9.66e−3
1e−4	1284	24.7	9.97e−8	1e4	4000	74.2	2.66e−5
1e−3	1113	21.1	9.97e−8	1e3	3500	65.6	2.4e−6
1e−2	881	16.6	9.81e−8	1e2	2500	46.2	3.38e−7
1e−1	773	13.9	9.93e−8	1e1	1500	26.9	9.35e−8

In summary, due to its low iteration cost and effective usage of the memory of first-order information, the FAPL method is a powerful tool for solving ball-constrained smooth CP problems especially when the number of variables is huge and/or the value of Lipschitz constant is large. And by incorporating the proposed Expansion Algorithm, the unconstrained FAPL method is very efficient for solving unconstrained CP problems especially when a proper estimation on the optimal solution set is not available.

4.2 Total variation based image reconstruction

In this subsection, we apply the FUSL method to solve the nonsmooth total variation (TV) based image reconstruction problem:

$$\min_{u\in\mathbb{R}^N} \frac{1}{2}\|Au-b\|_2^2 + \lambda\|u\|_{TV}, \tag{4.2}$$

where A is a given matrix, u is the vector form of the image to be reconstructed, b represents the observed data, and $\|\cdot\|_{TV}$ is the discrete TV semi-norm defined by

$$\|u\|_{TV} := \sum_{i=1}^{N} \|D_i u\|_2, \tag{4.3}$$

where $D_i u \in \mathbb{R}^2$ is the discrete gradient (finite differences along the coordinate directions) of the i^{th} component of u, and N is the number of pixels in the image. Note that $\|u\|_{TV}$ is convex and nonsmooth.

One of the approaches to solve this problem is to consider the associated dual or primal-dual formulations of (4.3) based on the dual formulation of the TV norm:

$$\begin{aligned}\|u\|_{TV} = \max_{p\in Y} \langle p, Du\rangle, \text{ where } Y = \{p = (p_1,\ldots,p_N) \in \mathbb{R}^{2N} : p_i \in \mathbb{R}^2,\\ \|p_i\|_2 \le 1, 1 \le i \le N\}.\end{aligned} \tag{4.4}$$

Consequently, we can rewrite (4.2) as a saddle-point problem:

$$\min_{u\in\mathbb{R}^N} \max_{p\in Y} \frac{1}{2}\|Au-b\|_2^2 + \lambda \langle p, Du\rangle. \tag{4.5}$$

Note that (4.5) is exactly the form we considered in the USL and FUSL methods if we set $\hat{g}(y) = 0$. Specifically, the prox-function $v(y)$ on Y is simply chosen as $v(y) = \frac{1}{2}\|y\|^2$ in these smoothing techniques.

In our experiments, we consider two types of instances depending on how the matrix A is generated. Specifically, for the first case, the entries of A are normally distributed, while for the second one, the entries are uniformly distributed. For both types of instances, first, we generate the matrix $A \in \mathbb{R}^{m\times n}$, then choose a true image x_{ture} and convert it to a vector, and finally compute b by $b = Ax_{true} + \epsilon$, where

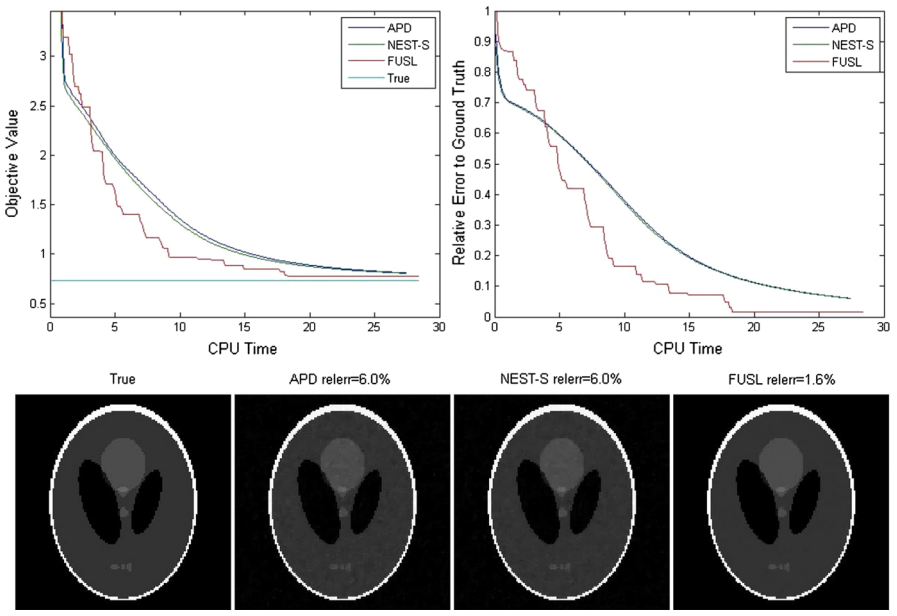

Fig. 1 TV-based reconstruction (Shepp–Logan phantom)

ϵ is the Gaussian noise with distribution $\epsilon = N(0, \sigma)$. We compare the following algorithms: the accelerated primal dual (APD) method [8], Nesterov's smoothing (NEST-S) method [5,34], and the FUSL method.

For our first experiment, the entries of the matrix A of size $4,096 \times 16,384$ is randomly generated from a normal distribution $N(0, 64)$, the image x_{true} is a 128×128 Shepp–Logan phantom generated by MATLAB. Moreover, we set $\lambda = 10^{-3}$ and the standard deviation $\sigma = 10^{-3}$. The values of Lipschitz constants are provided for APD and NEST-S, and the initial lower bound for FUSL is set to 0. We run 300 iterations for each algorithm, and present the objective values of problem (4.2) and the relative errors defined by $\|x_k - x_{true}\|_2/\|x_{true}\|_2$ in Fig. 1. In our second experiment, the matrix A is randomly generated with entries uniformly distributed in [0, 1]. We use a 200×200 brain image [38] as the true image x_{true}, and set $m = 20,000$, $\lambda = 10$, $\sigma = 10^{-2}$. Other setup is the same as the first experiment, and the results are shown in Fig. 2.

We make some observations about the results in Figs. 1 and 2. For the first experiment, there is almost no difference between APD and NEST-S, but FUSL outperforms both of them after 5 seconds in terms of both objective value and relative error. The second experiment clearly demonstrates the advantage of FUSL for solving CP problems with large Lipschitz constants. The Lipschitz constant of matrix A in this instance is about 2×10^8, much larger than the Lipschitz constant (about 5.9) in the first experiment. FUSL still converges quickly and decreases the relative error to 0.05 in less than 100 iterations, while APD and NEST-S converge very slowly and more than 1,000 iterations are required due to the large Lipschitz constants. It seems that FUSL is not so sensitive to the Lipschitz constants as the other two methods. This feature of

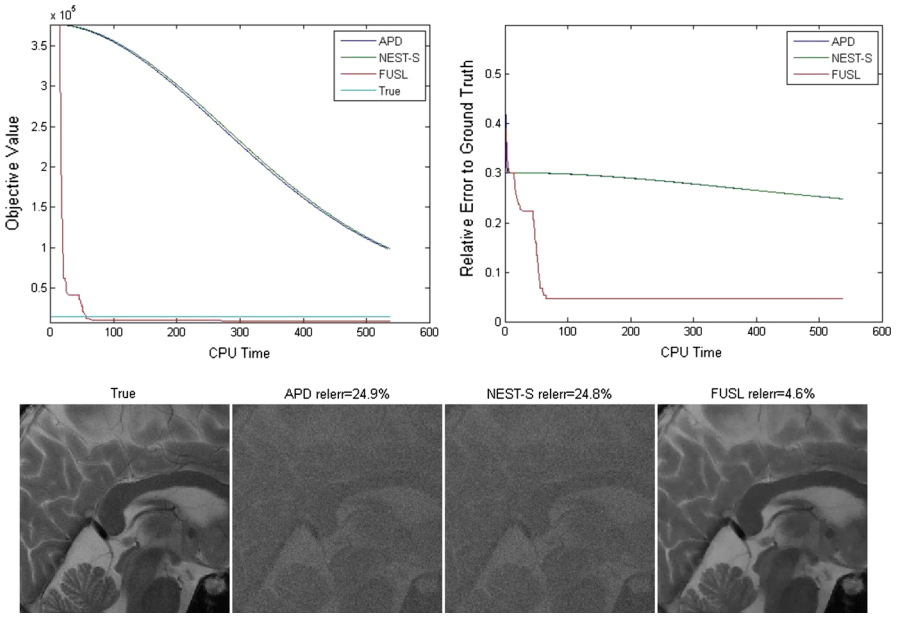

Fig. 2 TV-based reconstruction (brain image)

FUSL makes it more efficient for solving large-scale CP problems which often have big Lipschitz constants.

In summary, for the TV-based image reconstruction problem (4.2), FUSL not only enjoys the completely parameter-free property (and hence no need to estimate the Lipschitz constant), but also demonstrates advantages for its speed of convergence and its solution quality in terms of relative error, especially for large-scale problems.

4.3 Partially parallel imaging

In this subsection, we compare the performance of the FUSL method with several related algorithms in reconstruction of magnetic resonance (MR) images from partial parallel imaging (PPI), to further confirm the observations on advantages of the FUSL method. The detailed background and description of PPI reconstruction can be found in [38]. This image reconstruction problem can be modeled as

$$\min_{u\in C^n} \sum_{j=1}^{k} \|M\mathcal{F}S_j u - f_j\|^2 + \lambda \sum_{i=1}^{N} \|D_i u\|_2,$$

where $n = 2$ (we consider two dimensional case), u is the N-vector form of a two-dimensional complex valued image to be reconstructed, k is the number of coils (consider them as sensors) in the magnetic resonance (MR) parallel imaging system. $F \in C^{n\times n}$ is a 2D discrete Fourier transform matrix, $S_j \in C^{n\times n}$ is the sensitivity map of the j-th sensor, and $M \in R^{n\times n}$ is a binary mask describes the scanning pattern.

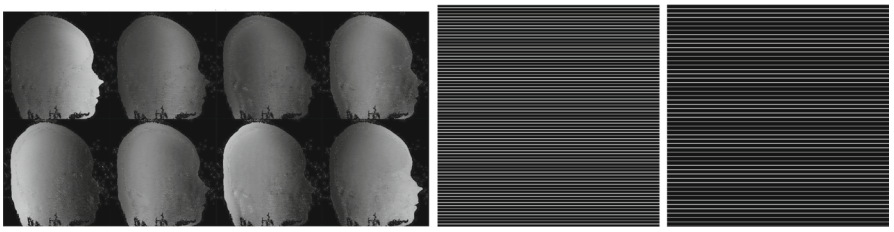

Fig. 3 Sensitivity map and Cartesian masks

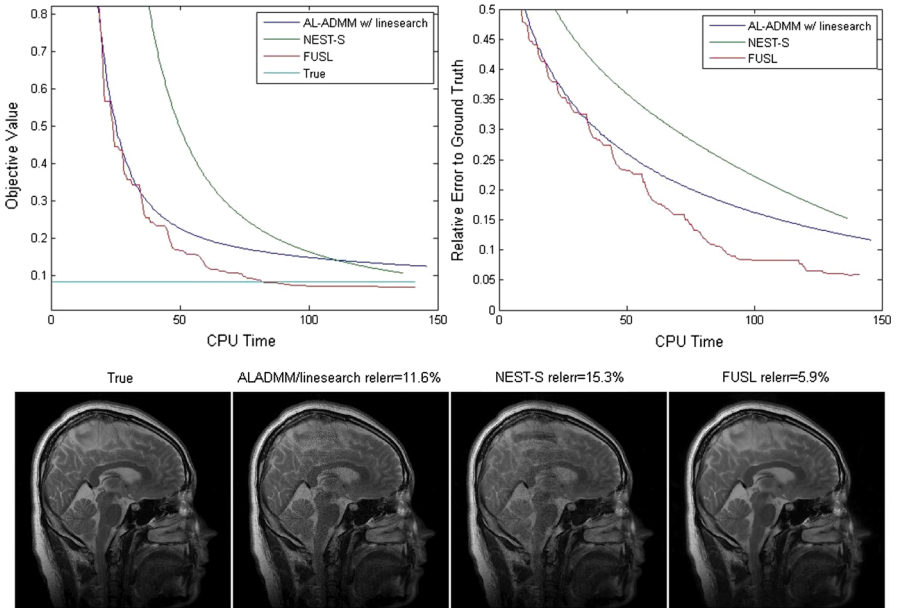

Fig. 4 PPI image reconstruction (acquisition rate: 14%)

Note that the percentages of nonzero elements in M describes the compression ratio of PPI scan. In our experiments, the sensitivity maps $\{S_j\}_{j=1}^k$ are shown in Fig. 3, the image x_{true} is of size 512×512 shown in Figs. 4 and 5, and the measurements $\{f_j\}$ are generated by

$$f_j = M(FS_j x_{true} + \epsilon_j^{re}/\sqrt{2} + \epsilon_j^{im}/\sqrt{-2}), \ \ j = 1, \ldots, k, \tag{4.6}$$

where $\epsilon_j^{re}, \epsilon_j^{im}$ are the noise with entries independently distributed according to $N(0, \sigma)$. We conduct two experiments on this data set with different acquisition rates, and compare the FUSL method to NEST-S method, and the accelerated linearized alternating direction of multipliers (AL-ADMM) with line-search method [15].

For both experiments, set $\sigma = 3 \times 10^{-2}$, $\lambda = 10^{-5}$, and $\{f_j\}_{j=1}^k$ are generated by (4.6). In the first experiment, we use Cartesian mask with acquisition rate 14%:

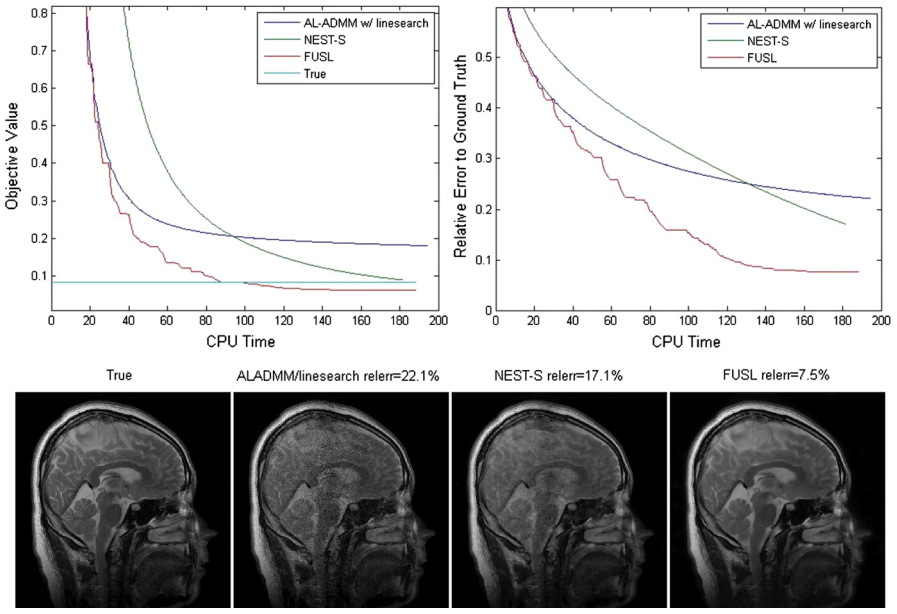

Fig. 5 PPI image reconstruction (acquisition rate: 10%)

acquire image in one row for every successive seven rows, while for the second one, we use Cartesian mask with acquisition rate 10%: acquire image in one row for every successive ten rows. The two masks are shown in Fig. 3. The results of the first and second experiments are shown in Figs. 4 and 5, respectively. These experiments again demonstrate the advantages of the FUSL method over other techniques for PPI image reconstruction,

5 Concluding remarks

In this paper, we presented two new BL type methods, the FAPL and FUSL methods, to uniformly solve smooth, nonsmooth, and weakly smooth CP problems and a class of structured nonsmooth problems with optimal iteration complexities. Because of the use of cutting plane model, technique of restricted memory and an efficient and scalable solver for solving involved subproblem, the proposed methods have lower iteration cost, and can find a solution with higher accuracy within less number of iterations than many gradient descent type methods. Moreover, these BL type methods do not require the input of any problem parameter, or involve any stepsize that could be affected by large Lipschitz constant of the objective function and large dimension of the problem. These built-in features of the proposed methods are essential for solving large-scale CP problems. Our numerical results of least squares problems and total variation based image reconstructions clearly demonstrate the advantages of the FAPL and FUSL methods over the original APL, USL and some other first-order methods.

Appendix A. Solving the subproblems of FAPL and FUSL

In this section, we introduce an efficient method to solve the subproblems (2.6) in the FAPL and FUSL methods, which are given in the form of

$$x_c^* := \operatorname{argmin}_{x \in Q} \frac{1}{2}\|x - p\|^2. \tag{A.1}$$

Here, Q is a closed polyhedral set described by m linear inequalities, i.e.,

$$Q := \{x \in \mathbb{R}^n : \langle A_i, x\rangle \le b_i,\ i = 1, 2, \ldots, m\},$$

where $A_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ for $1 \le i \le m$.

Now let us examine the Lagrange dual of (A.1) given by

$$\max_{\lambda \ge 0} \min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - p\|^2 + \sum_{i=1}^{m} \lambda_i [\langle A_i, x\rangle - b_i]. \tag{A.2}$$

It can be checked from the theorem of alternatives that problem (A.2) is solvable if and only if $Q \neq \emptyset$. Indeed, if $Q \neq \emptyset$, it is obvious that the optimal value of (A.2) is finite. On the other hand, if $Q = \emptyset$, then there exists $\bar{\lambda} \ge 0$ such that $\bar{\lambda}^T A = 0$ and $\bar{\lambda}^T b < 0$, which implies that the optimal value of (A.2) goes to infinity. Moreover, if (A.2) is solvable and λ^* is one of its optimal dual solutions, then

$$x_c^* = p - \sum_{i=1}^{m} \lambda_i^* A_i. \tag{A.3}$$

It can also be easily seen that (A.2) is equivalent to

$$\max_{\lambda \ge 0} -\frac{1}{2}\lambda^T M \lambda + C^T \lambda, \tag{A.4}$$

where $M_{ij} := \langle A_i, A_j \rangle$, $C_i := \langle A_i, p\rangle - b_i$, $\forall i, j = 1, 2, \ldots, m$. Hence, we can determine the feasibility of (A.1) or compute its optimal solution by solving the relatively simple problem (A.4).

Many algorithms are capable of solving the above nonnegative quadratic programming in (A.4) efficiently. Due to its low dimension (usually less than 10 in our practice), we propose a brute-force method to compute the exact solution of this problem. Consider the Lagrange dual associated with (A.4):

$$\min_{\lambda \ge 0} \max_{\mu \ge 0} \mathcal{L}(\lambda, \mu) := \frac{1}{2}\lambda^T M \lambda - (C^T + \mu)\lambda,$$

where the dual variable is $\mu := (\mu_1, \mu_2, \ldots, \mu_m)$. Applying the KKT condition, we can see that $\lambda^* \ge 0$ is a solution to problem (A.4) if and only if there exists $\mu^* \ge 0$ such that

$$\nabla_\lambda \mathcal{L}(\lambda^*, \mu^*) = 0 \quad \text{and} \quad \langle \lambda^*, \mu^* \rangle = 0. \tag{A.5}$$

Note that the first identity in (A.5) is equivalent to a linear system:

$$\begin{pmatrix} M & -I \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \\ \mu_1 \\ \vdots \\ \mu_m \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_m \end{pmatrix}, \tag{A.6}$$

where I is the $m \times m$ identity matrix. The above linear system has $2m$ variables and m equations. But for any $i = 1, \ldots, m$, we have either $\lambda_i = 0$ or $\mu_i = 0$, and hence we only need to consider 2^m possible cases on the non-negativity of these variables. Since m is rather small in practice, it is possible to exhaust all these 2^m cases to find the exact solution to (A.5). For each case, we first remove the m columns in the matrix $(M \ -I)$ which correspond to the m variables assumed to be 0, and then solve the remaining determined linear system. If all variables of the computed solution are non-negative, then solution (λ^*, μ^*) to (A.5) is found, and the exact solution x_c^* to (A.1) is computed by (A.3), otherwise, we continue to examine the next case. It is interesting to observe that these different cases can also be considered in parallel to take the advantages of high performance computing techniques.

References

1. Lan, G.: Bundle-level type methods uniformly optimal for smooth and nonsmooth convex optimization. Math. Program. **149**(1–2), 1–45 (2015)
2. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D: Nonlinear Phenom. **60**(2), 259–268 (1992)
3. Osher, S., Burger, M., Goldfarb, D., Jinjun, X., Yin, W.: An iterative regularization method for total variation-based image restoration. Multiscale Modeling Simul. **4**(2), 460–489 (2005)
4. Nemirovski, A.S., Yudin, D.: *Problem complexity and method efficiency in optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley, XV (1983)
5. Nesterov, Y.E.: Smooth minimization of nonsmooth functions. Math. Program. **103**, 127–152 (2005)
6. Nesterov, Y.E.: A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. Dokl. AN USSR. **269**, 543–547 (1983)
7. Nesterov, Y.E.: Introductory Lectures on Convex Optimization: a Basic Course. Kluwer Academic Publishers, Massachusetts (2004)
8. Chen, Y., Lan, G., Ouyang, Y.: Optimal primal-dual methods for a class of saddle point problems. SIAM J. Optim. **24**(4), 1779–1814 (2014)
9. He, Y., Monteiro, R.D.C.: An accelerated hpe-type algorithm for a class of composite convex-concave saddle-point problems. SIAM J. Optim. **26**(1), 29–56 (2016)
10. Hong, M., Luo, Z.Q.: On the linear convergence of the alternating direction method of multipliers. Math. Program. **162**, 1–35 (2012)
11. Deng, W., Yin, W.: On the global and linear convergence of the generalized alternating direction method of multipliers. J. Sci. Comput. **66**(3), 889–916 (2016)
12. Goldfarb, D., Ma, S., Scheinberg, K.: Fast alternating linearization methods for minimizing the sum of two convex functions. Math. Program. **141**(1–2), 349–382 (2013)

13. Monteiro, R.D.C., Svaiter, B.F.: Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. SIAM J. Optim. **23**(1), 475–507 (2013)
14. Goldstein, T., O'Donoghue, B., Setzer, S., Baraniuk, R.: Fast alternating direction optimization methods. SIAM J. Imaging Sci. **7**(3), 1588–1623 (2014)
15. Ouyang, Y., Chen, Y., Lan, G., Pasiliao Jr., E.: An accelerated linearized alternating direction method of multipliers. SIAM J. Imaging Sci. **8**(1), 644–681 (2015)
16. Kelley, J.E.: The cutting plane method for solving convex programs. J. SIAM. **8**, 703–712 (1960)
17. Kiwiel, K.C.: An aggregate subgradient method for nonsmooth convex minimization. Math. Program. **27**, 320–341 (1983)
18. Lemaréchal, C.: An extension of davidon methods to non-differentiable problems. Math. Program. Study. **3**, 95–109 (1975)
19. Kiwiel, K.C.: Proximal level bundle method for convex nondifferentable optimization, saddle point problems and variational inequalities. Math. Program. Ser. B. **69**, 89–109 (1995)
20. Lemaréchal, C., Nemirovski, A.S., Nesterov, Y.E.: New variants of bundle methods. Math. Program. **69**, 111–148 (1995)
21. Kiwiel, K.C.: Proximity control in bundle methods for convex nondifferentiable minimization. Math. Program. **46**, 105–122 (1990)
22. Ben-Tal, A., Nemirovski, A.S.: Non-Euclidean restricted memory level method for large-scale convex optimization. Math. Program. **102**, 407–456 (2005)
23. van Ackooij, W., Sagastizábal, C.: Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. SIAM J. Optim. **24**(2), 733–765 (2014)
24. de Oliveira, W., Sagastizábal, C., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM J. Optim. **21**(2), 517–544 (2011)
25. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex proximal bundle methods in depth: a unified analysis for inexact oracles. Math. Program. **148**(1–2), 241–277 (2014)
26. Richtárik, P.: Approximate level method for nonsmooth convex minimization. J. Optim. Theory Appl. **152**(2), 334–350 (2012)
27. Kiwiel, K.C.: A proximal bundle method with approximate subgradient linearizations. SIAM J. optim. **16**(4), 1007–1023 (2006)
28. Kiwiel, Krzysztof C: Bundle methods for convex minimization with partially inexact oracles. Comput. Optim. Appl., available from the web site SemanticScholar
29. de Oliveira, W, Sagastizábal, C: Level bundle methods for oracles with on-demand accuracy. Optim. Methods Softw. (ahead-of-print): 29,1–30 (2014)
30. Kiwiel, K.C., Lemaréchal, C.: An inexact bundle variant suited to column generation. Math. program. **118**(1), 177–206 (2009)
31. Brännlund, U., Kiwiel, K.C., Lindberg, P.O.: A descent proximal level bundle method for convex nondifferentiable optimization. Op. Res. Lett. **17**(3), 121–126 (1995)
32. Cruz, J.B., de Oliveira, W.: Level bundle-like algorithms for convex optimization. J. Glob. Optim. **59**, 1–23 (2013)
33. de Oliveira, W., Sagastizábal, C.: Bundle methods in the xxist century: a bird's-eye view. Pesqui. Op. **34**(3), 647–670 (2014)
34. Becker, S., Bobin, J., Candès, E.J.: Nesta: a fast and accurate first-order method for sparse recovery. SIAM J. Imaging Sci. **4**(1), 1–39 (2011)
35. Astorino, A., Frangioni, A., Gaudioso, M., Gorgone, E.: Piecewise-quadratic approximations in convex numerical optimization. SIAM J. Optim. **21**(4), 1418–1438 (2011)
36. Ouorou, A.: A proximal cutting plane method using chebychev center for nonsmooth convex optimization. Math. Program. **119**(2), 239–271 (2009)
37. Mosek. The mosek optimization toolbox for matlab manual. version 6.0 (revision 93). http://www.mosek.com
38. Chen, Y., Hager, W., Huang, F., Phan, D., Ye, X., Yin, W.: Fast algorithms for image reconstruction with application to partially parallel mr imaging. SIAM J. Imaging Sci. **5**(1), 90–118 (2012)

Affiliations

Yunmei Chen[1] · Guanghui Lan[2] · Yuyuan Ouyang[3] · Wei Zhang[1]

Yunmei Chen
yun@math.ufl.edu

Yuyuan Ouyang
yuyuano@clemson.edu

Wei Zhang
weizhang657@ufl.edu

1 Department of Mathematics, University of Florida, Gainesville, USA

2 H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, USA

3 Department of Mathematical Sciences, Clemson University, Clemson, USA