FAST BUNDLE-LEVEL TYPE METHODS FOR UNCONSTRAINED AND BALL-CONSTRAINED CONVEX OPTIMIZATION*

YUNMEI CHEN†, GUANGHUI LAN ‡, YUYUAN OUYANG §, AND WEI ZHANG ¶

Abstract. It has been shown in [14] that the 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 problems without requiring the 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. These hindered the applicability of these algorithms in solving large-scale and unconstrained optimization problems. In this paper, we first present a generic algorithmic framework to extend these uniformly optimal level methods for solving unconstrained problems. Moreover, we introduce two new variants of level methods, i.e., the fast APL (FAPL) method and the fast USL (FUSL) method, for solving large scale black-box and structured convex programming problems respectively. 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. Moreover, we present an exact method to solve the only subproblem for these algorithms. As a result, the proposed FAPL and FUSL methods have improved the performance of the APL and USL in practice significantly in terms of both computational time and solution quality. Our numerical results on solving some large-scale least square problems and total variation based image reconstruction have shown great advantages of these new bundle-level type methods over APL, USL, and some other state-of-the-art first-order methods.

Keywords: convex programming, first-order, optimal method, bundle-level type method, total variation, image reconstruction

AMS 2000 subject classification: 90C25, 90C06, 90C22, 49M37

1. Introduction.

Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, the main problem of interest in this paper is:

$$f^* := \min_{x \in \mathbb{R}^n} f(x). \tag{1.1}$$

Throughout this paper, we assume that the solution set X^* of (1.1) is nonempty. Moreover, denoting

$$x^* := \text{argmin}_x \{\|\overline{x} - x\| : x \in X^*\} \quad \text{and} \quad D^* := \|\overline{x} - x^*\| \tag{1.2}$$

for a given initial point $\overline{x} \in \mathbb{R}^n$, we assume in the black-box setting that for all closed sets

$$\Omega \subseteq B(\overline{x}, 4D^*) := \{x \in \mathbb{R}^n : \|x - \overline{x}\| \le 4D^*\}, \tag{1.3}$$

there exists $M(\Omega) > 0$ and $\rho(\Omega) \in [0, 1]$, such that[1]

$$f(y) - f(x) - \langle f'(x), y - x\rangle \le \frac{M(\Omega)}{1 + \rho(\Omega)} \|y - x\|^{1+\rho(\Omega)}, \quad \forall x, y \in \Omega. \tag{1.4}$$

Here $\|\cdot\|$ denotes the Euclidean norm, and $f'(x) \in \partial f(x)$, where $\partial f(x)$ denotes the subdifferential of f at $x \in \Omega$. Clearly, the above assumption on $f(\cdot)$ covers non-smooth ($\rho(\Omega) = 0$), smooth ($\rho(\Omega) = 1$) and weakly smooth ($0 < \rho(\Omega) < 1$) functions.

Our main goal in this paper is to develop a bundle-level method that is able to compute an approximate solution to problem (1.1), with uniformly optimal iteration complexity for non-smooth, smooth and weakly smooth objective functions (see [14]). Here, the iteration complexity is described by the number of evaluations of subgradients/gradients of f. In addition, we aim to design the bundle-level method so that its computational cost in each iteration is small, in order to improve its practical performance. Let us start by reviewing a few existing bundle-level methods.

*December, 2014. This research was partially supported by NSF grants CMMI-1254446, DMS-1319050, and ONR grant N00014-13-1-0036.

†Department of Mathematics, University of Florida (yun@math.ufl.edu).

‡Department of Industrial and System Engineering, University of Florida (glan@ise.ufl.edu).

§Department of Industrial and System Engineering, University of Florida (ouyang@ufl.edu) .

¶Department of Mathematics, University of Florida (weizhang657@ufl.edu).

[1]Observe that this assumption is weaker than requiring (1.4) holds for any $x, y \in \mathbb{R}^n$.

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 [8]. Consider the convex programming (CP) problem of

$$\min_{x\in X} f(x), \tag{1.5}$$

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 in X. In particular, given $x_1, x_2, \ldots, x_k \in X$, we approximate f by

$$m_k(x) := \max\{h(x_i, x), 1 \le i \le k\}, \tag{1.6}$$

and compute the iterates x_{k+1} by:

$$x_{k+1} \in \mathrm{Argmin}_{x\in X} m_k(x), \tag{1.7}$$

where

$$h(z, x) := f(z) + \langle f'(z), x - z\rangle\,. \tag{1.8}$$

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 [18, 20]. Some important improvements of Kelley's method have been made in 1990s under the name of bundle methods (see, e.g., [9, 10, 16]). In particular, by incorporating the level sets into Kelley's method, Lemaréchal, Nemirovskii and Nesterov [16] 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 basic 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^* 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

$$x_{k+1} = \mathrm{argmin}_{x\in X_k} \|x - x_k\|^2. \tag{1.9}$$

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^*, and consequently both L_k and X_k will converge to the set of optimal solutions for problem (1.5). It is shown in [16] the number of BL iterations required to find an ϵ-solution of (1.5), i.e., a point $\hat{x} \in X$ s.t. $f(\hat{x}) - f^* \le \epsilon$, can be bounded by $\mathcal{O}(1/\epsilon^2)$, which is optimal for general nonsmooth convex optimization.

Observe that for the above BL methods, the localizer X_k accumulates constraints, and hence that 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 [10, 3]. In particular, Ben-Tal and Nemirovski [3] 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 (1.9) is replaced by a general Bregman distance $d(\cdot)$ for exploiting the geometry of the feasible set X. NERML is often regarded as state-of-the-art for large-scale nonsmooth convex optimization as it substantially outperforms subgradient type methods in practice. Some more recent development of inexact proximal bundle methods and BL methods could be found in [27, 23, 22, 26, 12, 11, 7, 13].

1.2. Accelerated bundle-level type methods. While the classic BL method was optimal for solving nonsmooth CP problems only, Lan [14] recently significantly generalized this method so that they can optimally solve any black-box CP problems, including non-smooth, smooth and weakly smooth CP problems. In particular, for problem (1.5) with compact feasible set X, the two new BL methods proposed in [14], 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^* is estimated from the cutting plane model m_k in (1.6), the upper bound on f^* is given by the best objective value found so far, and the prox-center is updated by (1.9). 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 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 can also employ non-Euclidean prox-functions to explore the geometry of the feasible set X.

We now provide a brief description of the APL method. This method consists of multiple phases, and each phase calls a gap reduction procedure to reduce the gap between the lower and upper bounds on f^* by a constant factor. More specifically, at phase s, given the initial lower bound lb_s and upper bound ub_s, the APL gap reduction procedure sets $\underline{f}_0 = \mathrm{lb}_s, \overline{f}_0 = \mathrm{ub}_s, l = \beta \cdot \mathrm{lb}_s + (1-\beta)\mathrm{ub}_s$, and iteratively performs the following steps.

a) Set $x_k^l = (1-\alpha_k)x_{k-1}^u + \alpha_k x_{k-1}$ and $\underline{f}_k := \max\{\underline{f}_{k-1}, \min\{l, \underline{h}_k\}\}$, where

$$\underline{h}_k := \min_{x \in X_{k-1}} h(x_k^l, x). \tag{1.10}$$

b) Update the prox-center x_k by

$$x_k = \mathrm{argmin}_{x \in \underline{X}_k} d(x), \tag{1.11}$$

where $\underline{X}_k := \{x \in X_{k-1} : h(x_k^l, x) \le l\}$.

c) Set $\overline{f}_k = \min\{\overline{f}_{k-1}, f(\tilde{x}_k)\}$, where $\tilde{x}_k^u = \alpha_k x_k + (1-\alpha_k)x_{k-1}^u$, and x_k^u is chosen as either $\tilde{x}_k^u$ or x_{k-1}^u such that $f(x_k^u) = \overline{f}_k$.

d) Choose X_k such that $\underline{X}_k \subseteq X_k \subseteq \overline{X}_k$, where $\overline{X}_k := \{x \in X : \langle \nabla d(x_k), x - x_k \rangle \ge 0\}$.

Here $\nabla d(\cdot)$ denotes the gradient of $d(\cdot)$. Observe that the parameters α_k and β are fixed a priori, and do not depend on any problem parameters. Moreover, the localizer X_k is chosen between $\underline{X}_k$ and $\overline{X}_k$, so that the numbers of linear constraints in the two subproblems (1.10) and (1.11) can be fully controlled. It is shown in [14] that for problem (1.5) with compact feasible set X, the APL method achieves the optimal iteration complexity for any smooth, weakly smooth and non-smooth convex functions. Moreover, by incorporating Nesterov's smoothing technique [21] into the APL method, Lan also presented in [14] 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 the input of any problem parameters (see Subsection 3.2 for more details).

1.3. Contribution of this paper. One crucial problem associated with most existing BL type methods, including APL and USL, is that each phase of these algorithms involves solving two optimization problems; first a linear programing problem to compute the lower bound, and then a constrained quadratic programing problem to update the prox-center or new iterate. In fact, the efficiency of these algorithms relies on the solutions of the two involved subproblems (1.10) and (1.11), 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 programing problem. Secondly, the convergence analysis of limited memory BL type methods (e.g., NERML, APL, and USL) relies on the assumption that the feasible set is compact. These issues have significantly hindered the applicability of existing BL type methods. Our contribution in this paper mainly consists of the following three aspects.

Firstly, we propose a novel bundle-level framework for unconstrained CP problems. The proposed framework solves unconstrained CP problems through solutions to a series of ball-constrained CPs. In particular, if there exists a uniformly optimal method (e.g., the APL and USL methods) that solves ball-constrained black-box or structured CPs, then the proposed algorithm solves unconstrained black-box or structured CPs

with optimal complexity without requiring the input of any problem parameters as well. 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 [24] for more details).

Secondly, in order to solve ball-constrained CPs, we propose two greatly simplified BL type methods, namely the FAPL and FUSL methods, which achieve the same optimal iteration complexity as the APL and USL methods respectively, and maintain all the nice features of those methods, but has greatly reduced computational cost per iteration. Such improvement has been obtained by the reduction and simplification of the subproblems that have to be solved in the APL and USL methods. More specifically, we show that the linear optimization subproblem for computing the lower bound can be eliminated and that the ball constraint can be removed from the quadratic subproblem by properly choosing the prox-functions. We also generalize both FAPL and FUSL methods for solving strongly convex optimization problems and show that they can achieve the optimal iteration complexity bounds.

Thirdly, we introduce a simple exact approach to solve the only subproblem in our algorithms. As mentioned earlier, the accuracy of the solutions to the subproblems is essential for the efficiency of all these BL type methods mentioned in Sections 1.1 and 1.2. By incorporating the proposed exact approach to solve the only subproblem in our algorithm, the accuracy of the FAPL and FUSL methods is significantly increased and the total number of the iterations required to compute an ϵ-solution is greatly decreased comparing with the original APL and USL methods and other first order methods. Also when the number of linear constraints is fixed and small, the computational cost for solving the only subproblem only linearly depends on the dimension of the problem, since the cost for computing the vector inner product will dominate that for solving a few auxiliary linear systems. This feature is very important for solving large-scale CP problems.

Finally, we present very promising numerical results for these new FAPL and FUSL methods applied to solve large-scale least square problems and total-variation based image reconstruction problems. These algorithms significantly outperform other BL type methods, gradient type methods, and even the powerful MATLAB solver for linear systems, especially when the dimension and/or the Lipschitz constant of the problem is large.

It should be noted that there exist some variants of bundle-level methods [4, 7, 2] for solving nonsmooth CP problems in which the computation of the subproblem to update the lower bound $\underline{f}_k$ is skipped, so that the feasible set X is allowed to be unbounded. In each iteration, these methods apply a level feasibility check recursively in order to find a proper level l_k and the associated level set X_k, and update $\underline{f}_k$ to l_k if the level set associated with l_k is empty. However, in these methods, repeatedly checking the emptiness of level sets associated with varied levels in each iteration may be very costly in practice, especially when the linear constraints in the level sets accumulate. Also, if the feasible set $X = \mathbb{R}^n$, then for any chosen level, the corresponding level set consisting of linear constraints is unlikely to be empty, which would result in the inefficiency for updating the lower bound. In [2], an alternative for updating the lower bound (or increasing the level) is introduced by comparing the distances from stability center to the newly generated iterate and the solution set. This approach requires some prior knowledge about the distance to the solution set, and a rough estimate for that may lead to incorrect lower bounds and improper choices of levels.

1.4. Organization of the paper. The paper is organized as follows. In Section 2, we present a general scheme to extend the optimal BL type methods for unconstrained convex optimization. In Section 3, 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. We also extend the FAPL and FUSL methods to strongly convex CP and structured CP problems in Section 4, following some unpublished developments by Lan in [15]. The applications and promising numerical results are presented in Section 5.

2. Solving unconstrained CP problems through ball-constrained CP. Our goal in this section is to present a generic algorithmic framework to extend the uniformly optimal constrained BL algorithms in [14] for solving unconstrained problems.

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 of

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

where $B(\overline{x}, R)$ is defined in (1.3). 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 iteration complexity associated with $\mathcal{A}(\overline{x}, R, \epsilon)$ is given by

$$N_{\overline{x},R,\epsilon} := \frac{C_1(\overline{x}, R, f)R^{\alpha_1}}{\epsilon^{\beta_1}} + \frac{C_2(\overline{x}, R, f)R^{\alpha_2}}{\epsilon^{\beta_2}}, \tag{2.2}$$

where $\alpha_1 \geq \beta_1 > 0, \alpha_2 \geq \beta_2 > 0$ and $C_1(\overline{x}, R, f), C_2(\overline{x}, R, f)$ are some constants that depend on f in (2.1). For example, if f is a smooth convex function, ∇f is Lipschitz continuous in $\mathbb{R}^n$ with constant L, i.e., (1.4) holds with $\rho(\mathbb{R}^n) = 1$ and $M(\mathbb{R}^n) = L$, and we apply the APL method to (2.1), then we have only one term with $\alpha_1 = 1$, $\beta_1 = 1/2$, and $C_1(\overline{x}, R, f) = cL$ in (2.2), where c is a universal constant. Observe that the two complexity terms in (2.2) will be useful for analyzing some structured CP problems in Section 3.2. 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 in $B(\overline{x}, R)$.

By utilizing the aforementioned ball-constrained CP algorithm and a novel guess and check procedure, we present a bundle-level type algorithm for unconstrained convex optimiations as follows.

Algorithm 1 Bundle-level type methods for unconstrained CP problems

Choose initial estimation $r_0 \leq \|\overline{x} - x^*\|$ and compute the initial gap $\Delta_0 := f(\overline{x}) - \min_{x \in B(\overline{x}, r_0)} h(\overline{x}, x)$.
For $k = 0, 1, 2, \ldots$,

1. Set $\overline{x}'_k = \mathcal{A}(\overline{x}, r_k, \Delta_k)$ and $\overline{x}''_k = \mathcal{A}(\overline{x}, 2r_k, \Delta_k)$.
2. If $f(\overline{x}'_k) - f(\overline{x}''_k) > \Delta_k$, update $r_k \leftarrow 2r_k$ and go to step 1.
3. Otherwise, let $\overline{x}^*_k = \overline{x}''_k$, $\Delta_{k+1} = \Delta_k/2$ and $r_{k+1} = r_k$.

Step 1 and Step 2 in Algorithm 1 constitute a loop to find a pair of solution $(\overline{x}'_k, \overline{x}''_k)$ satisfying $0 \leq f(\overline{x}'_k) - f(\overline{x}''_k) \leq \Delta_k$. 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 $r_k \geq D^*$, where D^* is defined in (1.2). For simplicity, we call it an expansion if we double the radius in Step 2, and each iteration may contain several expansions before updating the output solution $\overline{x}^*_k$ in step 3.

Before analyzing the rate of convergence for Algorithm 1, we discuss some important observations related to the aforementioned expansions.

LEMMA 2.1. *Suppose that $\overline{x}$ is a fixed point and $R > 0$ is a fixed constant. Let $\overline{x}_1$ and $\overline{x}_2$ be ϵ-solutions to problems*

$$f^*_1 := \min_{x \in B(\overline{x}, R)} f(x) \quad \text{and} \quad f^*_2 := \min_{x \in B(\overline{x}, 2R)} f(x), \tag{2.3}$$

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

$$f(\overline{x}_2) - f^* \leq \left(3 + \frac{2D^*}{R}\right)\epsilon, \tag{2.4}$$

where f^* and D^* are defined in (1.1) and (1.2) respectively.

Proof. Clearly, by definition, we have $\|\overline{x}_1 - \overline{x}\| \leq R$, $\|\overline{x}_2 - \overline{x}\| \leq 2R$, $0 \leq f(\overline{x}_1) - f^*_1 \leq \epsilon$, and $0 \leq f(\overline{x}_2) - f^*_2 \leq \epsilon$. It suffices to consider the case when $f^*_2 > f^*$ and $\|x^* - \overline{x}\| > 2R$, since otherwise (2.4) holds trivially. Suppose x^*_1 and x^*_2 are the solutions to the first and second problems in (2.3) respectively, let $\hat{x}$ be the intersection of the line segment (x^*, x^*_1) with the ball $B(\overline{x}, 2R)$, 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

$$f(\hat{x}) \leq (1 - \frac{R_1}{R_2})f(x^*_1) + \frac{R_1}{R_2}f(x^*), \tag{2.5}$$

which implies that

$$\frac{R_1}{R_2}[f(x_1^*) - f(x^*)] \leq f(x_1^*) - f(\hat{x}), \tag{2.6}$$

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

$$f(x_1^*) - f(x_2^*) = [f(x_1^*) - f(\overline{x}_1)] + [f(\overline{x}_1) - f(\overline{x}_2)] + [f(\overline{x}_2) - f(x_2^*)] \tag{2.7}$$

$$\leq 0 + \epsilon + \epsilon = 2\epsilon. \tag{2.8}$$

Combining the previous inequalities, we obtain

$$f(x_1^*) - 2\epsilon \leq f(x_2^*) \leq f(\hat{x}) \leq f(x_1^*), \tag{2.9}$$

which implies that $f(x_1^*) - f(\hat{x}) \leq 2\epsilon$. Using the previous conclusion (2.6), and the fact that $R_1 \geq R$ and $R_2 \leq D^* + R$, we have

$$f(x_1^*) - f(x^*) \leq \frac{2\epsilon R_2}{R_1} \leq \left(2 + \frac{2D^*}{R}\right)\epsilon.$$

Therefore,

$$f(\overline{x}_2) - f(x^*) \leq f(\overline{x}_1) - f(x^*) \leq [f(\overline{x}_1) - f(x_1^*)] + [f(x_1^*) - f(x^*)] \leq \left(3 + \frac{2D^*}{R}\right)\epsilon.$$

□

We are now ready to prove the iteration complexity of Algorithm 1 for solving the unconstrained CP problem in (1.1).

THEOREM 2.2. *Suppose that the number of evaluations of f' in one call to $\mathcal{A}(\overline{x}, R, \epsilon)$ is bounded by (2.2), and denote $\epsilon_k := f(\overline{x}_k^*) - f^*$ for the iterates $\{\overline{x}_k^*\}$ of Algorithm 1. Then we have*

a) $r_k < 2D^*$ for all k;

b) $\lim_{k\to\infty} \epsilon_k = 0$;

c) The total number of evaluations of f' performed by Algorithm 1 up to the k-th iteration is bounded by

$$\mathcal{O}\left(\frac{C_1(\overline{x}, 4D^*, f)(D^*)^{\alpha_1}}{\epsilon_k^{\beta_1}} + \frac{C_2(\overline{x}, 4D^*, f)(D^*)^{\alpha_2}}{\epsilon_k^{\beta_2}}\right), \tag{2.10}$$

where D^* is defined in (1.2).

Proof. We start by proving that $r_k < 2D^*$ for all k. From the description of Algorithm 1, we see that expansions occur if and only if $f(\overline{x}_k') - f(\overline{x}_k'') > \Delta_k$ at Step 2. Moreover, we can observe that $f(\overline{x}_k') - f(\overline{x}_k'') \leq \Delta_k$ if $x^* \in B(\overline{x}, r_k)$. This observation implies that $r_k < 2D^*$. Indeed, it is easy to see that the total number of expansions is bounded by

$$\overline{S}_1 := \left\lceil \log_2 \frac{D^*}{r_0} \right\rceil + 1. \tag{2.11}$$

To prove b), noting from the description of Step 3 and Lemma 2.1 that

$$\epsilon_k = f(\overline{x}_k'') - f^* \leq \left(3 + \frac{2D^*}{r_k}\right)\Delta_k. \tag{2.12}$$

Since the total number of expansions is bounded by $\overline{S}_1$, and 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 performed by $\mathcal{A}$ to find $\overline{x}_k^*$ is K. Now we can estimate the total number of evaluations of f' performed by the K executions. For any $0 \le j < K$, as $2^{\alpha_1} \ge 2^{\beta_1} > 1$ and $2^{\alpha_2} \ge 2^{\beta_2} > 1$, we have

$$N'_{j+1} \ge 2^{\beta_1} N'_j \text{ and } N''_{j+1} \ge 2^{\beta_2} N''_j \tag{2.13}$$

by the assumption of $N_{\overline{x},R,\epsilon}$, where $N_j = N'_j + N''_j$ denotes the bound on iteration number for the j^{th} execution, and N'_j, N''_j correspond with the first and second terms in (2.10) respectively. Then the total number of iterations is bounded by

$$N := \sum_{j=1}^{K}(N'_j + N''_j) \le N'_K \sum_{j=0}^{K-1}(2^{\beta_1})^{-j} + N''_K \sum_{j=0}^{K-1}(2^{\beta_2})^{-j} \tag{2.14}$$

$$< N'_K \sum_{j=0}^{+\infty} 2^{-\beta_1 j} + N''_K \sum_{j=0}^{+\infty} 2^{-\beta_2 j} \le \frac{1}{1-2^{-\beta_1}} N'_K + \frac{1}{1-2^{-\beta_2}} N''_K \tag{2.15}$$

$$\le \frac{(1+2^{\alpha_1})C_1(\overline{x}, 2r_k, f)}{1-2^{-\beta_1}} \cdot \frac{r_k^{\alpha_1}}{\Delta_k^{\beta_1}} + \frac{(1+2^{\alpha_2})C_2(\overline{x}, 2r_k, f)}{1-2^{-\beta_2}} \cdot \frac{r_k^{\alpha_2}}{\Delta_k^{\beta_2}}. \tag{2.16}$$

Combining the above inequality with (2.12), we have

$$N < \sum_{i=1}^{2} \frac{(1+2^{\alpha_i})C_i(\overline{x}, 2r_k, f)}{1-2^{-\beta_i}} \cdot \frac{r_k^{\alpha_i}(3+\frac{2D^*}{r_k})^{\beta_i}}{\epsilon_k^{\beta_i}}. \tag{2.17}$$

Since $\alpha_i \ge \beta_i > 0$, then $r_k^{\alpha_i}(3+\frac{2D^*}{r_K})^{\beta_i} = r_k^{\alpha_i-\beta_i}(3r_k+2D^*)^{\beta_i}$ for $i = 1, 2$ is monotonically increasing with respect to r_k, which, in view of the fact $r_k < 2D^*$ for any $k \ge 0$, then clearly implies

$$N < \sum_{i=1}^{2} \frac{(1+2^{\alpha_i})2^{\alpha_i+3\beta_i}C_i(\overline{x}, 4D^*, f)}{2^{\beta_i}-1} \cdot \frac{(D^*)^{\alpha_i}}{\epsilon_k^{\beta_i}}. \tag{2.18}$$

Hence the proof is complete. □

Note that to solve the unconstrained black-box CP problem (1.1), the termination criterions of most first-order algorithms are based on the residual of the gradient or gradient mapping, which would lead to different complexity analysis. To the best of our knowledge, without any prior information on D^*, there is no any termination criterion based on functional optimality gap that could guarantee the termination of algorithms for finding an ϵ-solution of (1.1). Comparing to Nesterov's optimal gradient method for unconstrained problems in [19], Algorithm 1 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 $\bar{\epsilon}_k := f(x_k) - f^*$ for each iterate x_k. For both methods the efficiency estimates involve D^*. Since Algorithm 1 extend methods for ball-constraint CP problems to solve (1.1), and the iterations in the expansions of Algorithm 1 could be regarded as a guess and check procedure to determine D^*, it is reasonable that the efficiency estimates are only provided for unexpansive steps which update $\overline{x}_k^*$.

Since in [14] the APL method, and its variant the USL method, have optimal iteration complexity for solving smooth, nonsmooth, weakly smooth CP problems and structured nonsmooth CP problems on compact feasible sets, Algorithm 1 could be incorporated to solve (1.1) by specifying feasible sets to a sequences of Euclidean balls. Therefore, the main problem remained is how to improve the efficiency of these BL type methods for solving ball-constrained CP problems.

3. Fast prox-level type methods for ball-constrained and unconstrained problems. 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 Subsection 3.1, and then present the fast USL

(FUSL) method for solving a special class of ball-constrained structured CP problems in Subsection 3.2. We show how to solve the subproblems in these two algorithms in Subsection 3.3. We also briefly discuss in Subsection 3.4 the applications of the FAPL and FUSL methods for unconstrained optimization, based on our results in Section 2. For the sake of simplicity, throughout this section, we denote $\rho = \rho(B(\overline{x}, R)), M = M(B(\overline{x}, R))$ for (2.1).

3.1. FAPL for ball-constrained black-box problems. Our goal in this subsection is to present the FAPL method, which can significantly 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 (1.10) and (1.11)) 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 multiple phases, and in each phase the FAPL gap reduction procedure, denoted by $\mathcal{G}_{FAPL}$, is called to reduce the gap between the upper and lower bounds on $f^*_{\overline{x},R}$ 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 feasible sets $\underline{Q}_k$ and $\overline{Q}_k$ (see steps 1 and 4) in procedure $\mathcal{G}_{FAPL}$ 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^* in the APL method. Instead, in the FAPL method, the lower bound is updated to l directly whenever $\underline{Q}_k = \emptyset$ or $\|x_k - \overline{x}\| > R$. Thirdly, we choose a specific prox-function $d(x) = \frac{1}{2}\|x - \overline{x}\|^2$, and the prox-center is fixed to be $\overline{x}$. As a result, all the three sequences $\{x_k\}, \{x_k^l\}$ and $\{x_k^u\}$ will reside in the ball $B(\overline{x}, R)$. At last, as we will show in next subsection, since the subproblem (3.4) only contains a limited number of linear constraints (depth of memory), we can solve it very efficiently, or even exactly if the depth of memory is small, say, ≤ 10.

We now add a few more remarks about the technical details of Procedure 1. Firstly, Procedure 1 is terminated at step 2 if $Q_k = \emptyset$ or $\|x_k - \overline{x}\| > R$, which can be checked automatically when solving the subproblem (3.4) (see Subsection 3.3 for more details). Secondly, in step 4, while 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 $\{x \in \mathbb{R}^n : \langle x_k - \overline{x}, x - x_k\rangle \geq 0\}$ and a few most recently generated half-spaces, each of which is defined by $\{x \in \mathbb{R}^n : h(x_\tau^l, x) \leq l\}$ 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 readily given in [14]:

$$\alpha_1 = 1,\ 0 < \alpha_k \leq 1,\ \gamma_k \|\tau_k(\rho)\|_{\frac{2}{1-\rho}} \leq c k^{-\frac{1+3\rho}{2}},\ \forall k \geq 1 \tag{3.8}$$

for some constants $c > 0$, where $\|\cdot\|_p$ denotes the l_p norm,

$$\gamma_k := \begin{cases} 1, & k = 1, \\ \gamma_{k-1}(1 - \alpha_k), & k \geq 2, \end{cases} \quad \text{and} \quad \tau_k(\rho) := \left\{ \frac{\alpha_1^{1+\rho}}{\gamma_1}, \frac{\alpha_2^{1+\rho}}{\gamma_2}, \ldots, \frac{\alpha_k^{1+\rho}}{\gamma_k} \right\}. \tag{3.9}$$

The following lemma, whose proof is given in [14], provides two examples for the selection of $\{\alpha_k\}$.

LEMMA 3.1.

a) If $\alpha_k = 2/(k+1)$, $k = 1, 2, \ldots$, then the condition (3.8) is satisfied with $c = 2^{1+\rho} 3^{-\frac{1-\rho}{2}}$.

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

$$\alpha_1 = \gamma_1 = 1, \alpha_k^2 = (1 - \alpha_k)\gamma_{k-1} = \gamma_k, \forall k \geq 2, \tag{3.10}$$

then the condition (3.8) holds with $c = 4/3^{\frac{1-\rho}{2}}$.

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

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 $x_0^u = \hat{x}$. Let $x_0 \in B(\overline{x}, R)$ be given arbitrarily.

1: Update the cutting plane model:

$$x_k^l = (1-\alpha_k)x_{k-1}^u + \alpha_k x_{k-1}, \tag{3.1}$$

$$h(x_k^l, x) = f(x_k^l) + \left\langle f'(x_k^l), x - x_k^l \right\rangle, \tag{3.2}$$

$$\underline{Q}_k = \{x \in Q_{k-1} : h(x_k^l, x) \le l\}. \tag{3.3}$$

2: Update the prox-center and lower bound:

$$x_k = \mathrm{argmin}_{x \in \underline{Q}_k} \left\{ d(x) := \frac{1}{2}\|x - \overline{x}\|^2 \right\}. \tag{3.4}$$

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

$$\tilde{x}_k^u = (1-\alpha_k)x_{k-1}^u + \alpha_k x_k, \tag{3.5}$$

$$x_k^u = \begin{cases} \tilde{x}_k^u, & \text{if } f(\tilde{x}_k^u) < \overline{f}_k, \\ x_{k-1}^u, & \text{otherwise,} \end{cases} \tag{3.6}$$

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

$$\overline{Q}_k := \{x \in \mathbb{R}^n : \langle x_k - \overline{x}, x - x_k \rangle \ge 0\}. \tag{3.7}$$

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

LEMMA 3.2. *Let $\mathcal{E}_f(l) := \{x \in B(\overline{x}, R) : f(x) \le l\}$. If $\mathcal{E}_f(l) \ne \emptyset$, then the following statements hold for procedure $\mathcal{G}_{FAPL}$.*

a) Step 4 is always well-defined unless procedure $\mathcal{G}_{FAPL}$ already terminated.

b) $\mathcal{E}_f(l) \subseteq \underline{Q}_k \subseteq Q_k \subseteq \overline{Q}_k$ for any $k \ge 1$.

c) If $\underline{Q}_k \ne \emptyset$, then problem (3.4) in step 2 has a unique solution. Moreover, if procedure $\mathcal{G}_{FAPL}$ terminates at step 2, then $l \le f^*$.

Proof. To prove part a), we will use induction to prove that $\mathcal{E}_f(l) \subseteq Q_k$ for all $k \ge 0$. Firstly, 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 $k \ge 1$, then from the definition of $\underline{Q}_k$ in (3.3) and the observation that $h(x_k^l, x) \le f(x) \le l$ for all $x \in \mathcal{E}_f(l)$, we have $\mathcal{E}_f(l) \subseteq \underline{Q}_k \subseteq Q_k$, hence part a) holds.

To prove b), it suffices to show that $\underline{Q}_k \subseteq \overline{Q}_k$, since Q_k is chosen between $\underline{Q}_k$ and $\overline{Q}_k$, and $\mathcal{E}_f(l) \subseteq \underline{Q}_k$ is proved from the above induction. By the definition of $\overline{Q}_k$ in (3.7), we have $\overline{Q}_k = \{x \in \mathbb{R}^n : d(x) \ge d(x_k)\}$, hence $\underline{Q}_k \subseteq \overline{Q}_k$, and part b) holds.

We now provide the proof of part c). From the definition of Q_k in step 4 and the definition of $\underline{Q}_k$ in (3.3) we can see that $\underline{Q}_k$ is the intersection of half-spaces, hence it is convex and closed. Therefore, the subproblem (3.4) 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 $\mathcal{E}_f(l) \ne \emptyset$, then by part b) 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, supposing that $\|x_k - \overline{x}\| > R$ and $\mathcal{E}_f(l) \ne \emptyset$, let $x_R^* := \mathrm{argmin}_{x \in B(\overline{x}, r)} f(x)$, it is clear that $x_R^* \in \mathcal{E}_f(l) \subseteq \underline{Q}_k$ by b), however $\|x_R^* - \overline{x}\| \le R < \|x_k - \overline{x}\|$ which contradicts the definition of x_k in (3.4). □

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 3.3. *Let* $\mathrm{ub} := f(\hat{x}), \mathrm{ub}^+ := f(x^+)$ *in procedure* $\mathcal{G}_{FAPL}$. *Whenever procedure* $\mathcal{G}_{FAPL}$ *terminates, we have* $\mathrm{ub}^+ - \mathrm{lb}^+ \leq q(\mathrm{ub} - \mathrm{lb})$, *where*

$$q := \max\{\beta, 1 - (1-\theta)\beta\}. \tag{3.11}$$

Proof. By the definition of x^u_k in (3.5) 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 $\mathrm{ub}^+ \leq \mathrm{ub}$. Procedure $\mathcal{G}_{FAPL}$ could terminate at either step 2 or 3. We first suppose that it terminates at step 2 after k iterations. Using the termination condition $\mathrm{lb}^+ = l = \beta \cdot \mathrm{lb} + (1-\beta)\mathrm{ub}$, we have

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

Now suppose that Procedure 1 terminates at step 3 after k iterations. We have $\mathrm{ub}^+ = \overline{f}_k \leq l + \theta(\mathrm{ub} - l)$ and $\mathrm{lb}^+ \geq \mathrm{lb}$. Using the fact that $l = \beta \cdot \mathrm{lb} + (1-\beta)\mathrm{ub}$, we conclude that

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

We conclude the lemma by combining the above two relations. ∎

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 [14].

PROPOSITION 3.4. *If the stepsizes* $\{\alpha_k\}_{k\geq 1}$ *are chosen such that* (3.8) *holds, then the number of iterations performed by procedure* $\mathcal{G}_{FAPL}$ *does not exceed*

$$N(\Delta) := \left(\frac{cMR^{1+\rho}}{(1+\rho)\theta\beta\Delta}\right)^{\frac{2}{1+3\rho}} + 1, \tag{3.12}$$

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

Proof. It can be easily seen from the definition of $\overline{Q}_k$ that $x_k = \mathrm{argmin}_{x\in\overline{Q}_k} d(x)$, which, in view of the fact that $Q_k \subseteq \overline{Q}_k$, then implies that $x_k = \mathrm{argmin}_{x\in Q_k} d(x)$. Using this observation, and the fact that $x_{k+1} \in Q_k$ due to (3.3) and (3.4), we have $\langle \nabla d(x_k), x_{k+1} - x_k\rangle \geq 0$. Since $d(x)$ is strongly convex with modulus 1, we have

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

Combining the above two relations, we conclude $\frac{1}{2}\|x_{k+1} - x_k\|^2 \leq d(x_{k+1}) - d(x_k)$. Summing up these inequalities for any $k \geq 1$, we conclude that

$$\frac{1}{2}\sum_{\tau=1}^{k}\|x_{\tau+1} - x_\tau\| \leq d(x_{k+1}) = \frac{1}{2}\|x_{k+1} - \overline{x}\|^2 \leq \frac{1}{2}R^2. \tag{3.13}$$

Now suppose that procedure $\mathcal{G}_{FAPL}$ does not terminate at the k^{th} iteration. Applying the relation (3.14) in [14], and notice that $\alpha_1 = 1$, we have

$$f(x^u_k) - l \leq \frac{M}{1+\rho}[2d(x_k)]^{\frac{1+\rho}{2}}\gamma_k\|\tau_k(\rho)\|_{\frac{2}{1-\rho}}. \tag{3.14}$$

In view of steps 2 and 3 in procedure 1, and using the fact that $l = \beta \cdot \mathrm{lb} + (1-\beta)\mathrm{ub}$ in step 0, we have

$$f(x^u_k) - l > \theta(\mathrm{ub} - l) = \theta\beta\Delta.$$

Combining the above two relations, and using (3.8) and (3.13), we obtain

$$\theta\beta\Delta < \frac{MR^{1+\rho}}{(1+\rho)} \cdot \frac{c}{k^{\frac{1+3\rho}{2}}}, \tag{3.15}$$

which implies that

$$k < \left(\frac{cMR^{1+\rho}}{(1+\rho)\theta\beta\Delta} \right)^{\frac{2}{1+3\rho}}. \tag{3.16}$$

□

In view of Lemma 3.3 and Proposition 3.4, 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 2 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 $p_1 \in \mathrm{Argmin}_{x \in B(\overline{x},R)} h(p_0, x)$, $\mathrm{lb}_1 = h(p_0, p_1), \mathrm{ub}_1 = \min\{f(p_0), f(p_1)\}$, 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 $\hat{x}_s$.
3: Set $(\hat{x}_{s+1}, \mathrm{lb}_{s+1}) = \mathcal{G}_{FAPL}(\hat{x}_s, \mathrm{lb}_s, R, \overline{x}, \beta, \theta)$ and $\mathrm{ub}_{s+1} = f(\hat{x}_{s+1})$.
4: Set $s = s + 1$ and go to step 2.

A phase of the FAPL method occurs whenever s increments by 1. For the sake of simplicity, each iteration of procedure $\mathcal{G}_{FAPL}$ is also referred to as an iteration of the FAPL method. The following theorem establishes the complexity bounds on the total numbers of phases and iterations performed by the FAPL method and its proof is similar to that of Theorem 4 in [14].

THEOREM 3.5. *If the stepsizes $\{\alpha_k\}$ in procedure $\mathcal{G}_{FAPL}$ are chosen such that (3.8) holds, then the following statements hold for the FAPL method.*

a) The number of phases performed by the FAPL method does not exceed

$$S := \left\lceil \max\left\{0, \log_{\frac{1}{q}} \left(\frac{(2R)^{1+\rho}M}{(1+\rho)\epsilon} \right)\right\} \right\rceil. \tag{3.17}$$

b) The total number of iterations performed by the FAPL method for computing an ϵ-solution of problem (2.1) can be bounded by

$$N(\epsilon) := S + \frac{1}{1 - q^{\frac{2}{1+3\rho}}} \left(\frac{cMR^{1+\rho}}{(1+\rho)\theta\beta\epsilon} \right)^{\frac{2}{1+3\rho}}, \tag{3.18}$$

where q is defined in (3.11).

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 the FAPL method and (1.4), we have

$$\Delta_1 \le f(p_1) - h(p_0, p_1) = f(p_1) - f(p_0) - \langle f'(p_0), p_1 - p_0 \rangle \le \frac{(2R)^{1+\rho}M}{1+\rho}. \tag{3.19}$$

Also, by Lemma 3.3 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 $\tilde{S}$ phases of the FAPL method, then we have

$$\Delta_S > \epsilon \ge \Delta_{S+1}. \tag{3.20}$$

Combining the above three inequalities, we conclude that

$$\epsilon < q^{S-1}\Delta_1 \le q^{S-1}\frac{(2R)^{1+\rho}M}{1+\rho}, \tag{3.21}$$

and part a) follows immediately from the above inequality. We are now ready to prove part b). In view of Lemma 3.3 and (3.20), we have $\Delta_s \ge \epsilon q^{s-S}$. Using this estimate, part a), and Proposition 3.4, we conclude that the total number of iterations performed by the FAPL method is bounded by

$$N(\epsilon) = \sum_{s=1}^{S} N_s = S + \left(\frac{cMR^{1+\rho}}{(1+\rho)\theta\beta}\right)^{\frac{2}{1+3\rho}} \sum_{s=1}^{S} q^{\frac{2(S-s)}{1+3\rho}} < S + \frac{1}{1-q^{\frac{2}{1+3\rho}}}\left(\frac{cMR^{1+\rho}}{(1+\rho)\theta\beta\epsilon}\right)^{\frac{2}{1+3\rho}}, \tag{3.22}$$

where N_s denotes the number of the iterations of phase s for $1 \le s \le S$. □

In view of Theorem 3.5, the FAPL method achieves the optimal iteration complexity bounds for solving nonsmooth, weakly smooth, and smooth CP problems, which are the same as the convergence properties of the ABL and APL method (see [18, 20] for the discussions on the complexity theories for solving CP problems, and [14] for the convergence properties of the ABL and APL methods).

3.2. FUSL for ball-constrained structured problems. In this subsection, we still consider the ball-constrained problem in (2.1), but assume that its objective function is given by

$$f(x) := \hat{f}(x) + F(x), \tag{3.23}$$

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

$$\hat{f}(y) - \hat{f}(x) - \langle \nabla \hat{f}(x), y - x\rangle \le \frac{L_{\hat{f}}}{2}\|y - x\|^2, \tag{3.24}$$

and

$$F(x) := \max_{y\in Y}\{\langle Ax, y\rangle - \hat{g}(y)\}. \tag{3.25}$$

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. Our goal is to present a new prox-level method for solving problem (2.1)-(3.23), which can significantly reduce the iteration cost of the USL method in [14].

Generally, the function F given by (3.25) is non-smooth. However, in an important work [21], Nesterov demonstrated that this function can be closely approximated by a class of smooth convex functions. In particular, letting $v : Y \to \mathbb{R}$ be a prox-function with modulus σ_v and denoting $c_v := \operatorname{argmin}_{v\in Y} v(y)$, we can approximate F in (3.25) by the smooth function

$$F_\eta(x) := \max_{y\in Y}\{\langle Ax, y\rangle - \hat{g}(y) - \eta V(y)\}, \tag{3.26}$$

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

$$V(y) := v(y) - v(c_v) - \langle \nabla v(c_v), y - c_v\rangle. \tag{3.27}$$

It was shown in [21] that the gradient of $F_\eta(\cdot)$ given by $\nabla F_\eta(x) = A^* y^*(x)$ is Lipschitz continuous with constant

$$L_\eta := \|A\|^2/(\eta\sigma_v), \tag{3.28}$$

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 in (3.26). Moreover, the "closeness" of $F_\eta(\cdot)$ to $F(\cdot)$ depends linearly on the smoothing parameter η, i.e.,

$$F_\eta(x) \le F(x) \le F_\eta(x) + \eta D_{v,Y}, \ \forall x \in X, \tag{3.29}$$

where

$$D_{v,Y} := \max_{y,z\in Y}\{v(y) - v(z) - \langle \nabla v(z), y - z\rangle\}. \tag{3.30}$$

Therefore, if we denote

$$f_\eta(x) := \hat{f}(x) + F_\eta(x), \tag{3.31}$$

then

$$f_\eta(x) \le f(x) \le f_\eta(x) + \eta D_{v,Y}. \tag{3.32}$$

Applying an optimal gradient method to minimize the smooth function f_η in (3.31), Nesterov proves in [21] that the iteration complexity for computing an ϵ-solution to problem (2.1)-(3.23) is bounded by $\mathcal{O}(1/\epsilon)$. 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 [21] into the APL method, Lan developed in [14] a new bundle-level type method, namely the uniform smoothing level (USL) method, to solve structured problems given in the form of (3.23). While the USL method achieves the same optimal iteration complexity as Nesterov's smoothing scheme in [21], 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 of two subproblems. Based on the USL method in [14] and our analysis of the FAPL method in Section 3.1, we propose a fast USL (FUSL) method that solves problem (3.23) 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 consists of different phases, and each phase calls 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) by a constant factor. 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 $k = 1$, $\overline{f}_0 = f(\hat{x}), l = \beta\cdot \mathrm{lb} + (1-\beta)\overline{f}_0$, $Q_0 = \mathbb{R}^n$, $x_0^u = \hat{x}$, $x_0 \in B(\overline{x}, R)$ be arbitrarily given, and

$$\eta := \theta(\overline{f}_0 - l)/(2D). \tag{3.33}$$

1: Update the cutting plane model: set x_k^l to (3.1), $\underline{Q}_k$ to (3.3), and

$$h(x_k^l, x) = h_\eta(x_k^l, x) = f_\eta(x_k^l) + \left\langle f'_\eta(x_k^l), x - x_k^l\right\rangle. \tag{3.34}$$

2: Update the prox-center: set x_k to (3.4). If $\underline{Q}_k = \emptyset$ or $\|x_k - \overline{x}\| > R$, then **terminate** with output $x^+ = x_{k-1}^u, D^+ = D, \mathrm{lb}^+ = l$.

3: Update the upper bound and the estimate of $D_{v,Y}$: set $\tilde{x}_k^u$ to (3.5), x_k^u to (3.6), and $\overline{f}_k = f(x_k^u)$. Check the following conditions:

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

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

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 (1.8) with the one for f_η (see (3.34)). Also note that for the USL method in [14], $\hat{f}$ 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 convex, and linearize both $\hat{f}$ and F_η in (3.34). 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 $D_{v,Y}$ in (3.30) and given as an input parameter to procedure $\mathcal{G}_{FUSL}$. Thirdly, same to the FAPL method, the parameters $\{\alpha_k\}$ are chosen according to (3.8). Such conditions are required to guarantee the optimal convergence of the FUSL method for solving problem (2.1)-(3.23). Fourthly, similar to the FAPL method, the feasible sets $\underline{Q}_k, Q_k, \overline{Q}_k$ only contains a limited number of linear constraints, and there is only one subproblem (i.e. (3.4)) 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 [14].

LEMMA 3.6. *The following statements hold for procedure $\mathcal{G}_{FUSL}$.*

a) If this procedure terminates at steps 2 or 3a), then we have $\mathrm{ub}^+ - \mathrm{lb}^+ \le q(\mathrm{ub} - \mathrm{lb})$, where q is defined in (3.11) and $\mathrm{ub} := \overline{f}_0, \mathrm{ub}^+ := f(x^+)$.

b) If this procedure terminates at step 3b), then $D < D_{v,Y}$ and $D^+ < 2D_{v,Y}$.

Proof. The proof of part a) is as the same as that of Lemma 3.3, and we only show part b) here. By the termination condition at step 3b), we have $f(x_k^u) > l + \theta(\mathrm{ub} - l)$ and $f_\eta(x_k^u) \le l + \frac{\theta}{2}(\mathrm{ub} - l)$. So,

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

We conclude from the above relation, (3.32), and (3.33) that

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

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

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

PROPOSITION 3.7. *Suppose that $\{\alpha_k\}_{k\ge1}$ in procedure $\mathcal{G}_{FUSL}$ are chosen such that (3.8) holds. Then, the number of iterations performed by this procedure does not exceed*

$$\overline{N}(\Delta, D) := R\sqrt{\frac{cL_{\hat{f}}}{\theta\beta\Delta}} + \frac{\sqrt{2}R\|A\|}{\theta\beta\Delta}\sqrt{\frac{cD}{\sigma_v}} + 1, \tag{3.35}$$

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

Proof. It is easy to see that the gradient of f_η in (3.23) has Lipschitz continuous gradient with constant $L = L_{\hat{f}} + L_\eta$, where L_η and $L_{\hat{f}}$ are defined in (3.28) and (3.24), respectively. Suppose that procedure $\mathcal{G}_{FUSL}$ does not terminate at step k. Noting that the prox-function $d(x)$ in procedure $\mathcal{G}_{FUSL}$ has modulus 1, similarly to the discussion on (3.14), we have

$$f_\eta(x_k^u) - l \le \frac{cLd(x_k)}{k^2} \le \frac{cLR^2}{2k^2}, \tag{3.36}$$

where c is defined in (3.8), and the second inequality is from (3.13). Also, since procedure $\mathcal{G}_{FUSL}$ does not terminate, in view of the termination condition at step 3b) and the definition of l in step 0, we have

$$f_\eta(x_k^u) - l > \frac{\theta\beta\Delta}{2}. \tag{3.37}$$

Combining the above two relations, and noting (3.28) and (3.33), we conclude that

$$k \le \sqrt{\frac{cLR^2}{\theta\beta\Delta}} \le R\sqrt{\frac{cL_{\hat{f}}}{\theta\beta\Delta}} + \frac{\sqrt{2}R\|A\|}{\theta\beta\Delta}\sqrt{\frac{cD}{\sigma_v}}. \tag{3.38}$$

∎

Algorithm 3 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 (3.26) and (3.27), initial guess D_1 on the size $D_{v,Y}$ in (3.30), 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.

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)-(3.23).

Similar to the FAPL method, we say that a phase 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 [21], then we can set $D_1 = D_{v,Y}$ in the scheme of both the original and modified FUSL method, and consequently, all the phases of both the original and modified FUSL methods become significant.

For the sake of simplicity, 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 of problem (2.1)-(3.23). Note that the proof of these results is similar to that of Theorem 7 in [14].

THEOREM 3.8. *Suppose that $\{\alpha_k\}$ in procedure $\mathcal{G}_{FUSL}$ are chosen such that (3.8) holds. Then, the total number of iterations performed by the FUSL method for computing an ϵ-solution of problem (2.1)-(3.23) is bounded by*

$$\overline{N}(\epsilon) := S_1 + S_2 + \Big(\frac{2}{\sqrt{2}-1} + \frac{\sqrt{2}}{1-q}\Big)\frac{R\|A\|}{\theta\beta\epsilon}\sqrt{\frac{c\tilde{D}}{\sigma_v}} + \Big(S_1 + \frac{1}{1-\sqrt{q}}\Big)R\sqrt{\frac{cL_{\hat{f}}}{\theta\beta\epsilon}}, \tag{3.39}$$

where q and $D_{v,Y}$ are defined in (3.11) and (3.30) respectively, and

$$\tilde{D} := \max\{D_1, 2D_{v,Y}\}, S_1 := \max\left\{\left\lceil \log_2 \frac{D_{v,Y}}{D_1}\right\rceil, 0\right\} \text{ and } S_2 := \left\lceil \log_{\frac{1}{q}} \frac{4\sqrt{2}R\|A\|\sqrt{\frac{D_{v,Y}}{\sigma_v}} + 2R^2L_{\hat{f}}}{\epsilon}\right\rceil. \tag{3.40}$$

Proof. We prove this result by estimating the numbers of iterations performed within both non-significant and significant phases. Suppose that the set 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 3.6, 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 in non-significant phases by

$$\begin{aligned}\overline{N}_1 &= \sum_{k=1}^{s_1} \overline{N}(\Delta_{m_k}, D_{m_k}) \le \sum_{k=1}^{s_1} \overline{N}\left(\epsilon, \tilde{D}/2^{s_1-k}\right) \\ &\le S_1\left(R\sqrt{\frac{cL_{\hat{f}}}{\theta\beta\epsilon}}+1\right) + \frac{\sqrt{2}R\|A\|}{\theta\beta\epsilon}\sqrt{\frac{c}{\sigma_v}}\sum_{k=1}^{S_1}\sqrt{\frac{\tilde{D}}{2^{S_1-k}}} \\ &\le S_1\left(R\sqrt{\frac{cL_{\hat{f}}}{\theta\beta\epsilon}}+1\right) + \frac{2R\|A\|}{(\sqrt{2}-1)\theta\beta\epsilon}\sqrt{\frac{c\tilde{D}}{\sigma_v}}.\end{aligned} \tag{3.41}$$

Applying Lemma 8 in [14] and relation (3.24), and in view of the fact that $p_0, p_1 \in B(\overline{x}, R)$ in Algorithm 3, the initial gap is bounded as

$$\Delta_1 := \mathrm{ub}_1 - \mathrm{lb}_1 \le [F(p_0) - F(p_1) - \langle F'(p_1), p_0 - p_1\rangle] + \left[\hat{f}(p_0) - \hat{f}(p_1) - \left\langle \hat{f}'(p_1), p_0 - p_1\right\rangle\right] \tag{3.42}$$

$$\le 4\sqrt{2}R\|A\|\sqrt{\frac{D_{v,Y}}{\sigma_v}} + 2R^2 L_{\hat{f}}, \tag{3.43}$$

where $F'(p_1) \in \partial F(p_1)$. Then for significant phases, similarly to the proof of Theorem 3.5, we have $s_2 \le S_2$. Moreover, for any n_k, $1 \le k \le s_2$, using Lemmas 3.3, 3.6, 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 $\Delta_{n_k} > \epsilon/q^{s_2-k}$. Combining such an estimate on D_{n_k}, Δ_{n_k} and bound on s_2, we can see that the total number of iterations performed the significant phases is bounded by

$$\begin{aligned}\overline{N}_2 &= \sum_{k=1}^{s_2} \overline{N}(\Delta_{n_k}, D_{n_k}) \le \sum_{k=1}^{s_2} \overline{N}(\epsilon/q^{s_2-k}, \tilde{D}) \\ &\le S_2 + R\sqrt{\frac{cL_{\hat{f}}}{\theta\beta\epsilon}}\sum_{k=1}^{S_2} q^{\frac{S_2-k}{2}} + \frac{\sqrt{2}R\|A\|}{\theta\beta\epsilon}\sqrt{\frac{c\tilde{D}}{\sigma_v}}\sum_{k=1}^{S_2} q^{S_2-k} \\ &\le S_2 + \frac{R}{1-\sqrt{q}}\sqrt{\frac{cL_{\hat{f}}}{\theta\beta\epsilon}} + \frac{\sqrt{2}R\|A\|}{\theta\beta\epsilon(1-q)}\sqrt{\frac{c\tilde{D}}{\sigma_v}}.\end{aligned} \tag{3.44}$$

Finally, the total number of iterations performed by the FUSL method is bounded by $\overline{N}_1 + \overline{N}_2$, and thus (3.39) holds. □

From (3.39) in the above theorem, we can see that the iteration complexity of the FUSL method for solving problem (2.1)-(3.23) is bounded by

$$\mathcal{O}\left(\sqrt{\frac{L_{\hat{f}}}{\epsilon}} + \frac{\|A\|}{\epsilon}\right). \tag{3.45}$$

The above iteration complexity is the same as that of the Nesterov smoothing scheme in [21] and the USL method in [14]. 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 3.3 below, the FUSL method further improves the USL method by significantly reducing its iteration cost and improving the accuracy for solving its subproblems.

3.3. Solving the subproblems of FAPL and FUSL. In this section, we introduce an efficient method to solve the subproblems (3.4) in the FAPL and FUSL methods, which are given in the form of

$$x_c^* := \mathrm{argmin}_{x\in Q}\frac{1}{2}\|x - p\|^2. \tag{3.46}$$

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

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

Now let us examine the Lagrange dual of (3.46) given by

$$\max_{\lambda \ge 0} \min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - p\|^2 + \sum_{i=1}^{m} \lambda_i[\langle A_i, x\rangle - b_i]. \tag{3.47}$$

It can be checked from the theorem of alternatives that problem (3.47) is solvable if and only if $Q \neq \emptyset$. Indeed, if $Q \neq \emptyset$, it is obvious that the optimal value of (3.47) 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 (3.47) goes to infinity. Moreover, if (3.47) is solvable and λ^* is one of its optimal dual solutions, then

$$x_c^* = p - \sum_{i=1}^{m} \lambda_i^* A_i. \tag{3.48}$$

It can also be easily seen that (3.47) is equivalent to

$$\max_{\lambda \ge 0} -\frac{1}{2}\lambda^T M \lambda + C^T \lambda, \tag{3.49}$$

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 (3.46) or compute its optimal solution by solving the relatively simple problem in (3.49).

Many algorithms are capable of solving the above nonnegative quadratic programming in (3.49) 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 (3.49):

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

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 of problem (3.49) if and only if there exists $\mu^* \ge 0$ such that

$$\nabla_\lambda \mathcal{L}(\lambda^*, \mu^*) = 0 \quad \text{and} \quad \langle \lambda, \mu\rangle = 0. \tag{3.50}$$

Note that the first identity in (3.50) is equivalent to a linear system:

$$\begin{pmatrix} M & -I \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \\ \mu_1 \\ \vdots \\ \mu_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \tag{3.51}$$

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 (3.50). 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 (3.50) is found, and the exact solution x_c^* to (3.46) is computed by (3.48), 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.

3.4. Extending FAPL and FUSL for unconstrained problems. In this subsection, we study how to utilize the FAPL and FUSL method to solve the unconstrained problems based on our results in Section 2.

Let us first consider the case when f in (1.1) satisfies (1.4). If the method $\mathcal{A}$ in step 1 of Algorithm 1 is given by the FAPL method, then by Theorem 3.5, the number of evaluations of f' within one call to $\mathcal{A}(\overline{x}, 2r_k, \Delta_k)$ is bounded by

$$\left[\frac{cM_k(2r_k)^{1+\rho_k}}{(1+\rho_k)\theta\beta\Delta_k}\right]^{\frac{2}{1+3\rho_k}}, \tag{3.52}$$

where c is a universal constant, $M_k := M(B(\overline{x}, 2r_k))$ and $\rho_k := \rho(B(\overline{x}, 2r_k))$ are constants corresponding to the assumption in (1.4). By Theorem 2.2, the number of evaluations of f' up to the k-th iteration of Algorithm 1 is bounded by

$$\mathcal{O}\left(\left[\frac{M(4D^*)^{1+\rho}}{\epsilon_k}\right]^{\frac{2}{1+3\rho}}\right), \tag{3.53}$$

where $M := M(B(\overline{x}, 4D^*))$, $\rho := \rho(B(\overline{x}, 4D^*))$ and $\epsilon_k := f(x_k) - f^*$ is the accuracy of the solution. It should be noted that the constants M and ρ are local constants that depend on the distance from $\overline{x}$ and x^*, which are not required for the FAPL method and Algorithm 1, 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 (1.1) is given in the form of (3.23) as a structured nonsmooth CP problem, then the FUSL method could be applied to solve the corresponding structured ball-constraint problem in Algorithm 1. By Theorem 3.8, the number of evaluations of f' within one call to $\mathcal{A}(\overline{x}, 2r_k, \Delta_k)$ is bounded by

$$S_1 + S_2 + 2r_k C'\sqrt{\frac{L_{\hat{f}}}{\Delta_k}} + \frac{2r_k C''\|A\|}{\Delta_k}, \tag{3.54}$$

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

Applying Theorem 2.2 with $\alpha_1 = \alpha_2 = 1$, $\beta_1 = \frac{1}{2}$, $\beta_2 = 1$, $C_1(\overline{x}, R, f) = 2C'\sqrt{L_{\hat{f}}}$, and $C_2(\overline{x}, R, f) = 2C''\|A\|$, the number of evaluations of f' up to the k-th iteration of Algorithm 1 is bounded by

$$\mathcal{O}\left(4D^*C'\sqrt{\frac{L_{\hat{f}}}{\epsilon_k}} + \frac{4C''D^*\|A\|}{\epsilon_k}\right). \tag{3.55}$$

Similar to the FAPL method, here $L_f := L_f(B(\overline{x}, 4D^*))$ is a lower bound of $L_f(\mathbb{R}^n)$.

4. Generalization to strongly convex optimization. In this section, we generalize the FAPL and FUSL methods for solving convex optimization problems in the form of (1.1) whose objective function f satisfies

$$f(y) - f(x) - \langle f'(x), y - x\rangle \geq \frac{\mu}{2}\|y - x\|^2, \quad \forall x, y \in \mathbb{R}^n, \tag{4.1}$$

for some $\mu > 0$. For the sake of simplicity, we assume throughout this section that an initial lower bound $\mathrm{lb}_0 \leq f^*$ is available[2]. Under this assumption, it follows from (4.1) that $\|p_0 - x^*\|^2 \leq 2[f(p_0) - \mathrm{lb}_0]/\mu$ for a given initial point p_0, and hence that the FAPL and FUSL methods for ball-constrained problems can be directly applied. However, since the lower and upper bounds on f^* are constantly improved in each phase of these algorithms, we can shrink the ball constraints by a constant factor once every phase accordingly. We show that the rate of convergence of the FAPL and FUSL methods can be significantly improved in this manner.

[2]Otherwise, we should incorporate a guess-and-check procedure similar to the one in Section 2.

We first present a modified FAPL method for solving black-box CP problems which satisfy both (1.4) and (4.1). More specifically, we modify the ball constraints used in the FAPL method by shifting the prox-center $\bar{x}$ and shrinking the radius R in procedure $\mathcal{G}_{FAPL}$. Clearly, such a modification does not incur any extra computational cost. This algorithm is formally described as follows.

Procedure 3 The modified FAPL gap reduction procedure: $(x^+, \mathrm{lb}^+) = \tilde{\mathcal{G}}_{FAPL}(\hat{x}, \mathrm{lb}, r, \beta, \theta)$

In Procedure 1, set $\overline{x} = \hat{x}$, and consequently the prox-function d in (3.4) is replaced by $\|x - \hat{x}\|^2/2$.

Algorithm 4 The modified FAPL method for minimizing strongly convex functions

In Algorithm 2, change steps 0, 1 and 3 to

0: Choose initial lower bound $\mathrm{lb}_1 \le f^*$, initial point $p_0 \in \mathbb{R}^n$, initial upper bound $\mathrm{ub}_1 = f(p_0)$, tolerance $\epsilon > 0$ and parameters $\beta, \theta \in (0,1)$.

1: Set $\hat{x}_1 = p_0$, and $s = 1$.

3: Set $(\hat{x}_{s+1}, \mathrm{lb}_{s+1}) = \tilde{\mathcal{G}}_{FAPL}(\hat{x}_s, \mathrm{lb}_s, \sqrt{2(f(\hat{x}_s) - \mathrm{lb}_s)/\mu}, \beta, \theta)$ and $\mathrm{ub}_{s+1} = f(\hat{x}_{s+1})$.

A few remarks on the above modified FAPL method are in place. Firstly, let x^* be the optimal solution of problem (1.1) and define $\Delta_s = \mathrm{ub}_s - \mathrm{lb}_s$. By the definition of ub_s and lb_s, we have $f(\hat{x}_s) - f(x^*) \le \Delta_s$, which, in view of (4.1), then implies that

$$\|\hat{x}_s - x^*\|^2 \le \frac{2\Delta_s}{\mu} =: r^2, \tag{4.2}$$

and $x^* \in B(\hat{x}_s, r)$. Secondly, similar to procedure $\mathcal{G}_{FAPL}$, if procedure $\tilde{\mathcal{G}}_{FAPL}$ terminates at step 2, we have $\mathcal{E}_f(l) \cap B(\hat{x}_s, r) = \emptyset$. Combining this with the fact $x^* \in B(\hat{x}_s, r)$, we conclude that l is a valid lower bound on f^*. Therefore, no matter whether procedure $\tilde{\mathcal{G}}_{FAPL}$ terminates at step 2 or step 4, the gap between upper and lower bounds on f^* has been reduced and $\Delta_{s+1} \le q\Delta_s$, where the q is defined in (3.11).

We establish in Theorem 4.1 the iteration complexity bounds of the modified FAPL method for minimizing strongly convex functions.

THEOREM 4.1. *Suppose that $\{\alpha_k\}_{k\ge1}$ in procedure $\tilde{\mathcal{G}}_{FAPL}$ are chosen such that (3.8) holds. Then the total number of iterations performed by the modified FAPL method for computing an ϵ-solution of problem (1.1) is bounded by*

$$\widetilde{S}\left(\sqrt{\frac{2cM}{\theta\beta\mu}} + 1\right) \quad \text{and} \quad \widetilde{S} + \frac{1}{1 - q^{\frac{1-\rho}{1+3\rho}}}\left(\frac{4^{1+\rho}cM}{\theta\beta(1+\rho)\mu^{\frac{1+\rho}{2}}\epsilon^{\frac{1-\rho}{2}}}\right)^{\frac{2}{1+3\rho}},$$

respectively, for smooth strongly convex functions (i.e., $\rho = 1$) and nonsmooth or weakly smooth strongly convex functions (i.e., $\rho \in [0,1)$), where q is defined in (3.11), lb_1 and ub_1 are given initial lower bound and upper bound on f^*, and

$$\widetilde{S} := \left\lceil \log_{\frac{1}{q}}\left(\frac{\mathrm{ub}_1 - \mathrm{lb}_1}{\epsilon}\right)\right\rceil. \tag{4.3}$$

Proof. Suppose that procedure $\tilde{\mathcal{G}}_{FAPL}$ does not terminate at the k^{th} inner iteration. It then follows from (3.14) and (4.2) that

$$f(x_k^u) - l \le \frac{Mr^{1+\rho}}{1+\rho} \cdot \frac{c}{k^{\frac{1+3\rho}{2}}}. \tag{4.4}$$

Moreover, in view of the termination condition at steps 3 and relation (4.2), we have $f(x_k^u) - l \geq \theta(\mathrm{ub}_s - l) = \theta\beta\Delta_s$ and $r = \sqrt{2\Delta_s/\mu}$. Combining all the above observations we conclude that

$$k \leq \left(\frac{2^{\frac{1+\rho}{2}} cM}{\theta\beta(1+\rho)\mu^{\frac{1+\rho}{2}} \Delta_s^{\frac{1-\rho}{2}}} \right)^{\frac{2}{1+3\rho}}. \tag{4.5}$$

So the number of inner iterations performed in each call to procedure $\tilde{\mathcal{G}}_{FAPL}$ is bounded by

$$\left(\frac{2^{\frac{1+\rho}{2}} cM}{\theta\beta(1+\rho)\mu^{\frac{1+\rho}{2}} \Delta_s^{\frac{1-\rho}{2}}} \right)^{\frac{2}{1+3\rho}} + 1. \tag{4.6}$$

Since the gap between the upper and lower bounds on f^* is reduced by a constant factor in each phase, i.e., $\Delta_{s+1} \leq q\Delta_s$, it easy to see that the total number of phases is bounded by $\tilde{S}$ defined above. Using the previous two conclusions and the fact that $\Delta_s \geq \epsilon/q^{\widetilde{S}-s}$, we can show that the total number of iterations performed by the modified FAPL method is bounded by

$$\widetilde{S} + \left(\frac{2^{\frac{1+\rho}{2}} cM}{\theta\beta(1+\rho)\mu^{\frac{1+\rho}{2}} \epsilon^{\frac{1-\rho}{2}}} \right)^{\frac{2}{1+3\rho}} \sum_{s=1}^{\widetilde{S}} q^{(\widetilde{S}-s)\frac{1-\rho}{1+3\rho}}. \tag{4.7}$$

Specifically, if f is smooth ($\rho = 1$), then the above bound is reduced to

$$\widetilde{S}\left(\sqrt{\frac{2cM}{\theta\beta\mu}} + 1 \right). \tag{4.8}$$

If f is nonsmooth ($\rho = 0$) or weakly smooth ($\rho \in (0,1)$), then the above bound is equivalent to

$$\widetilde{S} + \left(\frac{2^{\frac{1+\rho}{2}} cM}{\theta\beta(1+\rho)\mu^{\frac{1+\rho}{2}} \epsilon^{\frac{1-\rho}{2}}} \right)^{\frac{2}{1+3\rho}} \sum_{s=1}^{\widetilde{S}} q^{(\widetilde{S}-s)\frac{1-\rho}{1+3\rho}} \leq \widetilde{S} + \frac{1}{1-q^{\frac{1-\rho}{1+3\rho}}} \left(\frac{2^{\frac{1+\rho}{2}} cM}{\theta\beta(1+\rho)\mu^{\frac{1+\rho}{2}} \epsilon^{\frac{1-\rho}{2}}} \right)^{\frac{2}{1+3\rho}}. \tag{4.9}$$

□

Now let us consider the structured CP problems with f given by (3.23), where the smooth component $\hat{f}$ is strongly convex with modulus μ. Similar to the modified FAPL method, we present a modified FUSL method for solving this strongly convex structured CP problems as follows.

Procedure 4 The modified FUSL gap reduction procedure: $(x^+, D^+, \mathrm{lb}^+) = \tilde{\mathcal{G}}_{FAPL}(\hat{x}, D, \mathrm{lb}, r, \beta, \theta)$

In Procedure 2, set $\overline{x} = \hat{x}$, and consequently the prox-function d is replaced by $\|x - \hat{x}\|^2/2$.

Algorithm 5 The modified FUSL method for minimizing strongly convex functions

In Algorithm 3, change steps 0,1 and 3 to

0: Choose initial lower bound $\mathrm{lb}_1 \leq f^*$, initial point $p_0 \in \mathbb{R}^n$, initial upper bound $\mathrm{ub}_1 = f(p_0)$, prox-function $v(\cdot)$, initial guess D_1 on the size $D_{v,Y}$, tolerance $\epsilon > 0$ and parameters $\beta, \theta \in (0,1)$.

1: Set $\hat{x}_1 = p_0$, and $s = 1$.

3: Set $(\hat{x}_{s+1}, D_{s+1}, \mathrm{lb}_{s+1}) = \tilde{\mathcal{G}}_{FUSL}(\hat{x}_s, D_s, \mathrm{lb}_s, \sqrt{2(f(\hat{x}_s) - \mathrm{lb}_s)/\mu}, \beta, \theta)$ and $\mathrm{ub}_{s+1} = f(\hat{x}_{s+1})$.

In the following theorem, we describe the convergence properties of the modified FUSL method for solving (1.1)-(3.23) with strongly convex smooth component $\hat{f}$.

THEOREM 4.2. *Suppose that $\{\alpha_k\}_{k\geq 1}$ in procedure $\tilde{\mathcal{G}}_{FUSL}$ are chosen such that (3.8) holds. Then we have the following statements hold for the modified FUSL method.*

a) The total number of iterations performed by the modified FUSL method for computing an ϵ-solution of problem (1.1)-(3.23) is bounded by

$$(S_1+\widetilde{S})\left(\sqrt{\frac{2cL_{\hat{f}}}{\theta\beta\mu}}+1\right)+\frac{4\|A\|\sqrt{\tilde{D}}}{\theta\beta(1-\sqrt{q})}\sqrt{\frac{c}{\sigma_v\mu\epsilon}}, \tag{4.10}$$

where q is defined in (3.8), S_1 and $\tilde{D}$ are defined in (3.40), and $\widetilde{S}$ is defined in (4.3).

b) In particular, if $D_{v,Y}$ is known, and set $D_1 = D_{v,Y}$ at Step 0, then the number of iterations performed by the modified FUSL method is reduced to

$$\overline{N}(\epsilon) := \widetilde{S}\left(\sqrt{\frac{2cL_{\hat{f}}}{\theta\beta\mu}}+1\right)+\frac{2\|A\|}{\theta\beta(1-\sqrt{q})}\sqrt{\frac{cD_{v,Y}}{\sigma_v\mu\epsilon}}. \tag{4.11}$$

Proof. Similarly to the discussion in Theorem 3.8, we classify the non-significant and significant phases and estimates the numbers of iterations performed by each type of phases. Suppose that the set 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. Then the number of nonsignificant phases is bounded by S_1, i.e., $s_1 \le S_1$. And since $\Delta_1 = \mathrm{ub}_1 - \mathrm{lb}_1$, so the number of significant phases is bounded by $\widetilde{S}$ defined above, i.e., $s_2 \le S_2$.

In view of Proposition 3.7, and substitute $r = \sqrt{\frac{2\Delta}{\mu}}$, we have for any phase

$$\widetilde{N}(\Delta, D) := \sqrt{\frac{2cL_{\hat{f}}}{\theta\beta\mu}}+\frac{2\|A\|}{\theta\beta}\sqrt{\frac{cD}{\sigma_v\mu\Delta}}+1. \tag{4.12}$$

Following similar discussion in Theorem 3.8, we have the number of iterations performed by non-significant phases in the modified FUSL method is bounded by

$$\widetilde{N}_1 = \sum_{k=1}^{s_1}\widetilde{N}(\Delta_{m_k}, D_{m_k}) \le \sum_{k=1}^{s_1}\widetilde{N}(\epsilon, \tilde{D}/2^{s_1-k}) \tag{4.13}$$

$$\le S_1\left(\sqrt{\frac{2cL_{\hat{f}}}{\theta\beta\mu}}+1\right)+\frac{2\|A\|}{\theta\beta}\sqrt{\frac{c\tilde{D}}{\sigma_v\mu\epsilon}}\sum_{k=1}^{S_1}q^{\frac{S_1-k}{2}} \tag{4.14}$$

$$\le S_1\left(\sqrt{\frac{2cL_{\hat{f}}}{\theta\beta\mu}}+1\right)+\frac{2\|A\|}{\theta\beta(1-\sqrt{q})}\sqrt{\frac{c\tilde{D}}{\sigma_v\mu\epsilon}}. \tag{4.15}$$

And the bound on number of iterations performed by all significant phases is given by

$$\widetilde{N}_2 = \sum_{k=1}^{s_2}\widetilde{N}(\Delta_{n_k}, D_{n_k}) \le \sum_{k=1}^{s_2}\widetilde{N}(\epsilon/q^{s_2-k}, \tilde{D}) \tag{4.16}$$

$$\le \widetilde{S}\left(\sqrt{\frac{2cL_{\hat{f}}}{\theta\beta\mu}}+1\right)+\frac{2\|A\|}{\theta\beta}\sqrt{\frac{c\tilde{D}}{\sigma_v\mu\epsilon}}\sum_{k=1}^{\widetilde{S}}q^{\frac{\widetilde{S}-k}{2}} \tag{4.17}$$

$$\le \widetilde{S}\left(\sqrt{\frac{2cL_{\hat{f}}}{\theta\beta\mu}}+1\right)+\frac{2\|A\|}{\theta\beta(1-\sqrt{q})}\sqrt{\frac{c\tilde{D}}{\sigma_v\mu\epsilon}}. \tag{4.18}$$

Therefore, the total number of iterations is bounded by $\widetilde{N}_1 + \widetilde{N}_2$, and thus part a) holds.

For part b), in view of Lemma 3.6, we can see that if $D_1 = D_{v,Y}$, then $D_s \equiv D_{v,Y}$ for all $s \geq 1$, and all phases of the modified FUSL method are significant. Therefore, replace D_{n_k} and $\tilde{D}$ in (4.16), we can conclude part b) holds. □

In view of the above Theorem 4.2, we can see that the iteration complexity of the modified FUSL method for solving the structured CP problem (3.23) is bounded by $\mathcal{O}\left(\|A\|/\sqrt{\epsilon}\right)$.

5. Numerical experiments. In this section we present our experimental results of solving a few large-scale CP problems, including the quadratic programming problems with large Lipschitz constants, and two different types of variation based image reconstruction problems, using the FAPL and FUSL methods, and 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.

5.1. 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. For this purpose, we consider the quadratic programming problem:

$$\min_{\|x\| \leq 1} \|Ax - b\|^2, \tag{5.1}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We compare the FAPL method with Nesterov's optimal method (NEST) for smooth functions [21], NERML [3], and APL [14]. We also compare the FAPL method with the the built-in Matlab linear system solver in view of its good practical performance. In the APL method, the subproblems are solved by MOSEK [17], an efficient software package for linear and second-order cone programming. Two cases with different choices of the initial lower bound LB in this experiments are conducted: (1). $LB = 0$ and (2). $LB = -\infty$.

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 all the four methods to solve (5.1) with this set of data A and b, and the accuracy of the generated solutions are measured by $e_k = \|Ax_k - b\|^2$. The results are shown in Tables 5.1, 5.2 and 5.3.

The advantages of the FAPL method can be observed from these experiments. Firstly, it is evident that BL type methods have much less iterations than NEST especially when the Lipschitz constant is large. Among these three BL type methods, NERML requires much more iterations than APL and FAPL, which have optimal iteration complexity for this problem.

Secondly, compared with previous BL type methods (APL and NERML), FAPL has much lower computational cost for each iteration. The computational cost of FAPL method for each iteration is just slightly larger than that of NEST method. However, the cost of each iteration of APL and NERML is 10 times larger than that of NEST.

Thirdly, consider the difference of performance for setting the lower bound to be 0 and $-\infty$, it is also evident that FAPL method is more robust to the choice of the initial lower bound and it updates the lower bound more efficiently than the other two BL methods. Though setting the lower bound to $-\infty$ increases number of iterations for all the three BL method, a close examination reveals that the difference between setting the lower bound to zero and $-\infty$ for FAPL method is not so significant as that for APL and NERML methods, especially for large matrix, for example, the second one in Table 5.1 .

Fourthly, 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 significantly. 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}.

Finally, we can clearly see from Table 5.3 that FAPL is comparable to or significantly outperform 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 much improved by using C or FORTRAN implementation, which has been used in the MATLAB solver for linear systems.

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 smooth CP problems especially when the number of variables is huge and/or the value of Lipschitz constant is large.

5.2. Total-variation based image reconstruction. In this subsection, we apply the FUSL method to solve the non-smooth total-variation (TV) based image reconstruction problem:

$$\min_{u\in\mathbb{R}^N} \frac{1}{2}\|Au-b\|_2^2 + \lambda\|u\|_{TV}, \tag{5.2}$$

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

$$\|u\|_{TV} := \sum_{i=1}^{N} \|D_i u\|_2, \tag{5.3}$$

where $D_i u \in \mathbb{R}^2$ is a 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. The $\|u\|_{TV}$ is convex and non-smooth.

TABLE 5.1
Uniformly distributed QP instances

$A: n = 4000, m = 3000, L = 2.0e6, e_0 = 2.89e4$							
Alg	LB	Iter.	Time	Acc.	Iter.	Time	Acc.
FAPL	0	103	3.06	9.47e-7	142	3.76	8.65e-9
	$-\infty$	277	6.55	5.78e-7	800	19.18	2.24e-11
APL	0	128	37.10	9.07e-7	210	60.85	9.82e-9
	$-\infty$	300	85.65	6.63e-6	800	234.69	2.59e-9
NERML	0	218	58.32	9.06e-7	500	134.62	1.63e-8
	$-\infty$	300	84.01	1.02e-2	800	232.14	1.71e-3
NEST	-	10000	220.1	3.88e-5	20000	440.02	3.93e-6

$A: n = 8000, m = 4000, L = 8.0e6, e_0 = 6.93e4$							
Alg	LB	Iter.	Time	Acc.	Iter.	Time	Acc.
FAPL	0	70	4.67	7.74e-7	95	6.46	6.85e-10
	$-\infty$	149	8.99	6.27e-7	276	16.94	6.10e-10
APL	0	79	71.24	7.79e-7	144	129.52	3.62e-9
	$-\infty$	248	205.48	8.16e-7	416	358.96	8.68e-9
NERML	0	153	128.71	7.30e-7	300	251.79	4.03e-9
	$-\infty$	300	257.54	1.18e-3	800	717.13	9.24e-5
NEST	-	10000	681.03	5.34e-5	20000	1360.52	4.61e-6

FAPL method for large dimension matrix							
Matrix A:$m \times n$	LB	Iter.	Time	Acc.	Iter.	Time	Acc.
10000×20000	0	97	36.65	6.41e-11	185	69.31	7.29e-21
L=5.0e7	$-\infty$	207	73.70	8.28e-8	800	292.06	2.32e-15
10000×40000	0	67	49.95	9.21e-11	122	91.49	7.27e-21
L=1.0e8	$-\infty$	130	88.40	7.11e-8	421	295.15	1.95e-16
10000×60000	0	52	58.06	7.68e-11	95	106.14	8.43e-21
L=1.5e8	$-\infty$	156	160.93	9.84e-8	394	422.4	7.48e-16

One of the approaches to solve this problem is to consider the associated dual or primal-dual formulations of (5.3) based on the dual formulation of the TV norm:

$$\|u\|_{TV} = \max_{p \in Y} \langle p, Du \rangle \,, \text{where } Y = \{p = (p_1, \ldots, p_N) \in \mathbb{R}^{2N} : p_i \in \mathbb{R}^2, \|p_i\|_2 \leq 1, 1 \leq i \leq N\}. \tag{5.4}$$

Consequently, we can rewrite (5.2) as a saddle-point problem:

$$\min_{u \in \mathbb{R}^N} \max_{p \in Y} \frac{1}{2}\|Au - b\|_2^2 + \lambda \langle p, Du \rangle \,. \tag{5.5}$$

Note that (5.5) is exactly the form we considered in the USL and FUSL method if we let $\hat{g}(y) = 0$. 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 generated the matrix $A \in \mathbb{R}^{m \times n}$, then choose some true image x_{ture} and convert it to a vector, and finally compute b by $b = Ax_{true} + \epsilon$, where ϵ is the Gaussian noise with distribution $\epsilon = N(0, \sigma)$. We compare the following algorithms: the accelerated primal dual (APD) method [5], Nesterov's smoothing (NEST-S) method [21, 1], and FUSL method.

For our first experiment, the matrix A is randomly generated of size $4,096 \times 16,384$ with entries normally distributed according to $N(0, \sqrt{4,096})$, 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 Lipschitz constants are

TABLE 5.2
Gaussian distributed QP instances

$A: n = 4000, m = 3000, L = 2.32e4, e_0 = 2.03e3$							
Alg	LB	Iter.	Time	Acc.	Iter.	Time	Acc.
FAPL	0	105	2.78	8.43e-7	153	4.10	7.84e-10
	$-\infty$	338	8.02	6.86e-7	696	16.58	9.74e-10
APL	0	128	35.12	9.01e-7	172	47.49	9.28e-9
	$-\infty$	639	200.67	7.92e-7	800	258.25	1.03e-7
NERML	0	192	48.44	7.05e-7	276	70.31	1.09e-8
	$-\infty$	300	93.32	3.68e-1	800	257.25	6.41e-2
NEST	-	10000	211.30	7.78e-4	20000	422.78	1.95e-4

$A: n = 8000, m = 4000, L = 2.32e4, e_0 = 2.03e3$							
Alg	LB	Iter.	Time	Acc.	Iter.	Time	Acc.
FAPL	0	49	3.25	8.34e-7	68	4.37	7.88e-10
	$-\infty$	165	9.77	5.17e-7	280	16.18	5.06e-10
APL	0	59	48.91	8.59e-7	78	64.95	1.70e-8
	$-\infty$	300	268.47	9.81e-7	670	637.70	9.42e-10
NERML	0	105	181.23	9.14e-7	133	102.68	1.39e-8
	$-\infty$	300	282.56	9.92e-3	800	760.26	8.32e-4
NEST	-	10000	567.59	3.88e-4	20000	1134.38	9.71e-5

FAPL method for large dimension matrix							
Matrix A:$m \times n$	LB	Iter.	Time	Acc.	Iter.	Time	Acc.
10000×20000	0	78	27.88	7.22e-11	145	51.81	6.81e-21
L=5.7e4	$-\infty$	228	78.57	9.92e-8	800	280.19	1.37e-15
10000×40000	0	48	34.36	5.97e-11	87	62.24	8.26e-21
L=9e4	$-\infty$	156	106.12	7.18e-8	390	271.15	4.29e-16
10000×60000	0	34	36.30	9.88e-11	65	69.56	7.24e-21
L=1.2e5	$-\infty$	98	98.11	9.50e-8	350	361.83	8.34e-16

TABLE 5.3
Comparison to Matlab solver

Matrix A:$m \times n$	Matlab $A\backslash b$		FAPL method		
	Time	Acc.	Iter.	Time	Acc.
Uniform 2000×4000	4.41	5.48e-24	204	3.59	6.76e-23
Uniform 2000×6000	7.12	9.04e-24	155	4.10	9.73e-23
Uniform 2000×8000	9.80	9.46e-24	135	4.45	9.36e-23
Uniform 2000×10000	12.43	1.04e-23	108	4.23	7.30e-23
Gaussian 3000×5000	11.17	5.59e-25	207	6.25	7.18e-23
Gaussian 3000×6000	13.96	1.43e-24	152	5.50	9.59e-23
Gaussian 3000×8000	19.57	1.66e-24	105	4.83	8.17e-23
Gaussian 3000×10000	25.18	1.35e-24	95	5.43	5.81e-23

provided for APD and NEST-S, and the initial lower bound for FUSL method is set to 0. We run 300 iterations for all these algorithms, and report the objective value of problem (5.2) and the relative error defined by $\|x_k - x_{true}\|_2/\|x_{true}\|_2$ as shown in Figure 5.1. In our second experiment, the matrix A is randomly generated with entries uniformly distributed in $[0, 1]$. We use a 200×200 brain image [6] 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 Figure 5.2.

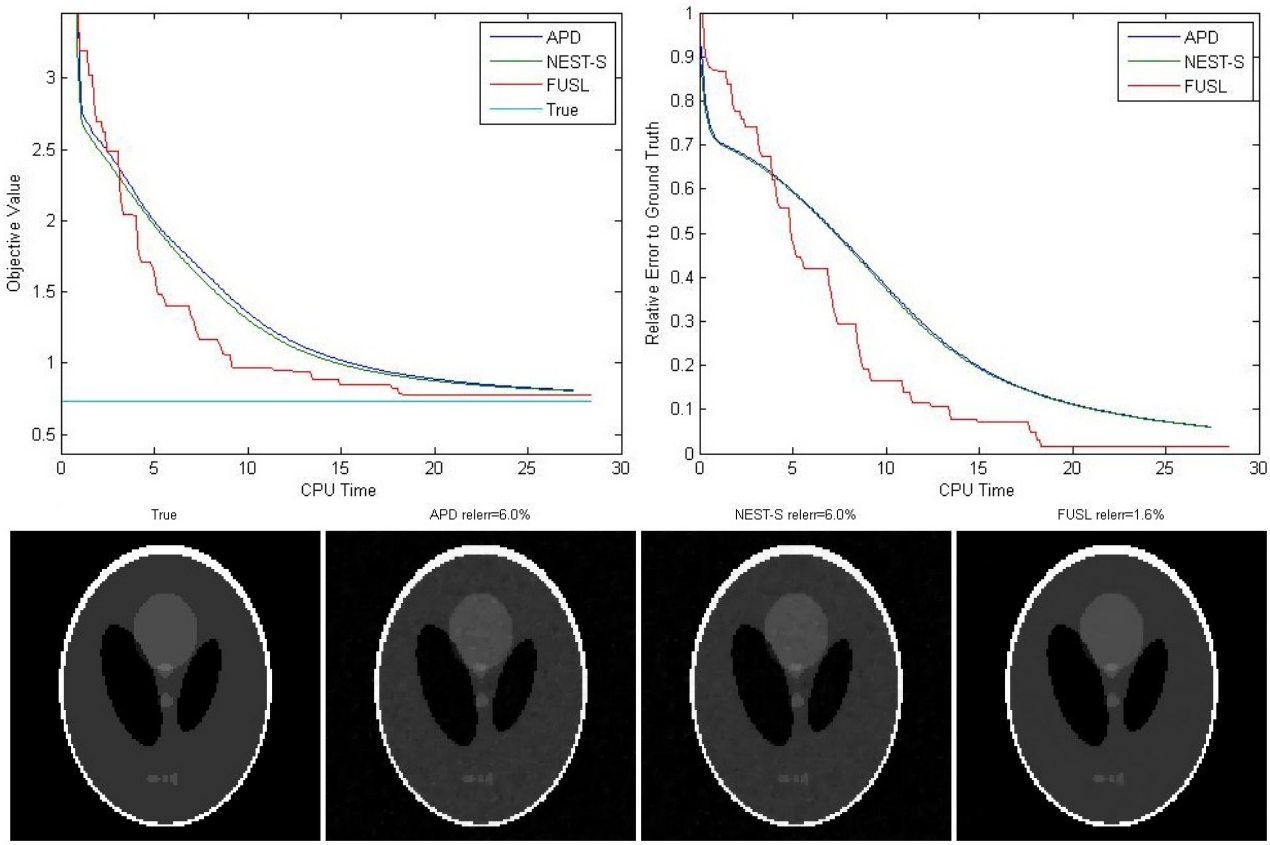

FIG. 5.1. *TV-based reconstruction (Shepp-Logan phantom)*

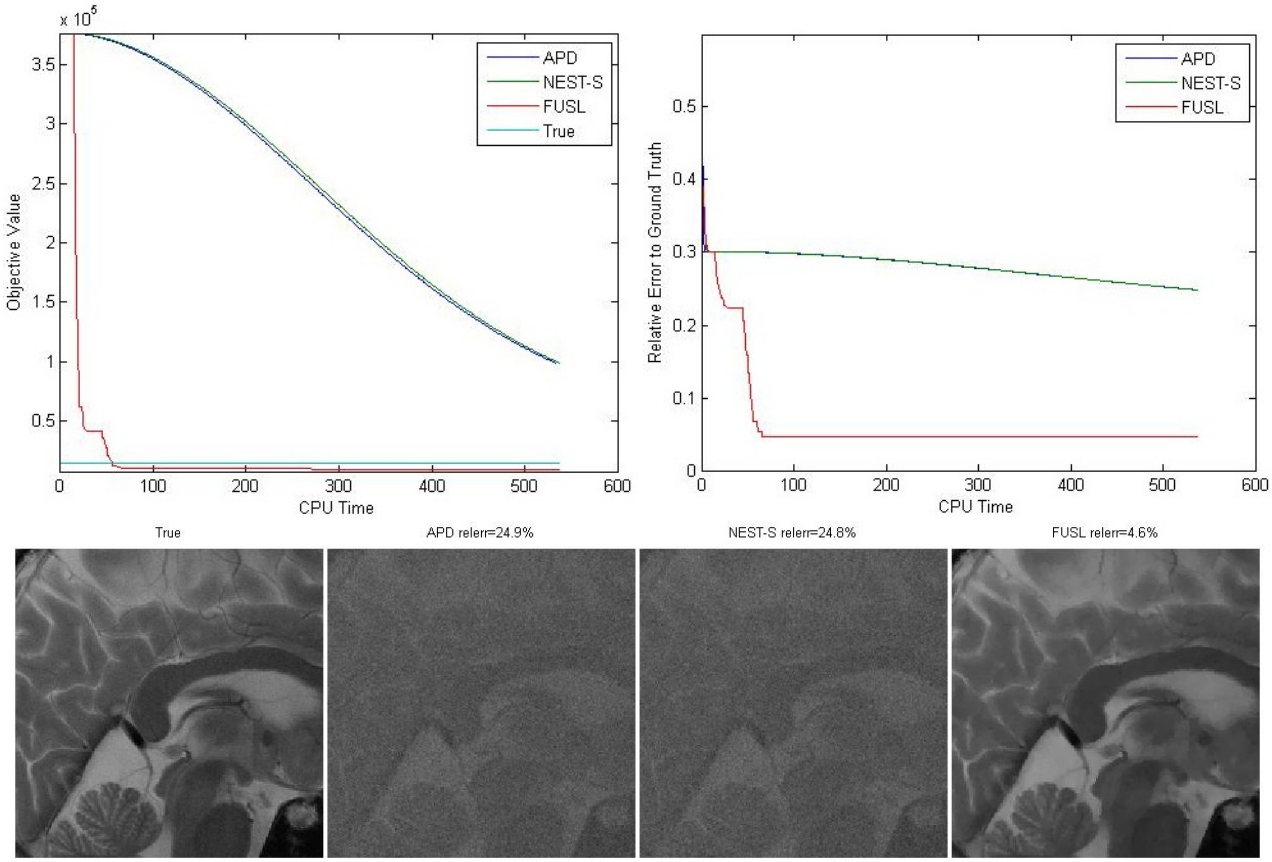

FIG. 5.2. *TV-based reconstruction (brain image)*

We make some observations about the results in Figures 5.1 and 5.2. For the first experiment, there is almost no difference between APD and NEST-S method, 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 $1,000$ steps 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 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 (5.2), FUSL not only enjoys the completely parameter-free property (and hence no need to estimate the Lipschitz constant), but also demonstrates significant advantages for its speed of convergence and its solution quality in terms of relative error, especially for large-scale problems.

5.3. 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 this method. The detailed background and description of PPI reconstruction can be found in [6]. This image reconstruction problem in two dimensional

cases 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 u is the vector form of a two-dimensional image to be reconstructed, k is the number of MR coils (consider them as sensors) in the 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. Note that the percentages of nonzero elements in M describes the compression ration of PPI scan. In our experiments, the sensitivity map $\{S_j\}_{j=1}^k$ is shown in Figure 5.3, the image x_{true} is of size 512×512 shown in Figures 5.4 and 5.5, and the measurements $\{f_j\}$ are generated by

$$f_j = M(FS_j x_{true} + \epsilon_j^{re}/\sqrt{2} + \epsilon_j^{im}/\sqrt{-2}),\ j = 1, \ldots, k, \tag{5.6}$$

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 [25].

For both experiments, set $\sigma = 3 \times 10^{-2}, \lambda = 10^{-5}$, and $\{f_j\}_{j=1}^k$ are generated by (5.6). In the first experiment, we use Cartesian mask with acquisition rate 14%: 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 Figure 5.3. The results of the first and second experiment are shown in Figures 5.4 and 5.5 respectively. These experiments again demonstrate the advantages of the FUSL method over these state-of-the-art techniques for PPI image reconstruction,

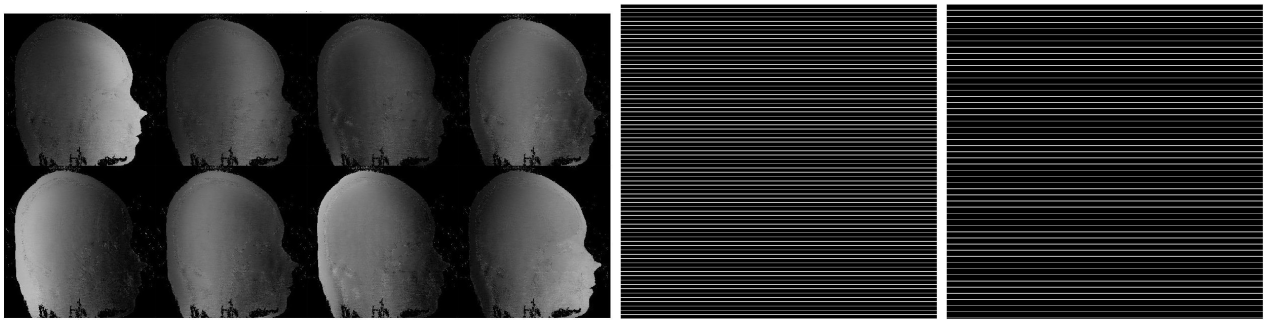

FIG. 5.3. *Sensitivity map and Cartesian mask*

6. Concluding remarks. In this paper, we propose two new bundle-level type methods, the FAPL and FUSL methods, to uniformly solve black-box smooth, nonsmooth, and weakly smooth CP problems and a class of structured nonsmooth problems. Our methods achieve the same optimal iteration complexity and maintain all the nice features of the original APL and USL methods. Meanwhile, by simplifying the subproblems involved in the algorithms and solving them exactly, the FAPL and FUSL methods reduce the iteration cost and increase the accuracy of solutions significantly, and thus overcome the drawback of these existing bundle-level type methods applied to large-scale CP problems. Furthermore, by introducing a generic algorithmic framework, we extend the uniformly optimal bundle-level type methods to unconstrained problems and broaden the applicability of these algorithms. The complexity of bundle-level type methods for unconstrained convex optimizations has been analyzed for the first time in the literature. The numerical results for least square problems and total variation based image reconstruction clearly demonstrate the advantages of FAPL and FUSL methods over the original APL, USL and some other state-of-the-art first-order methods.

REFERENCES

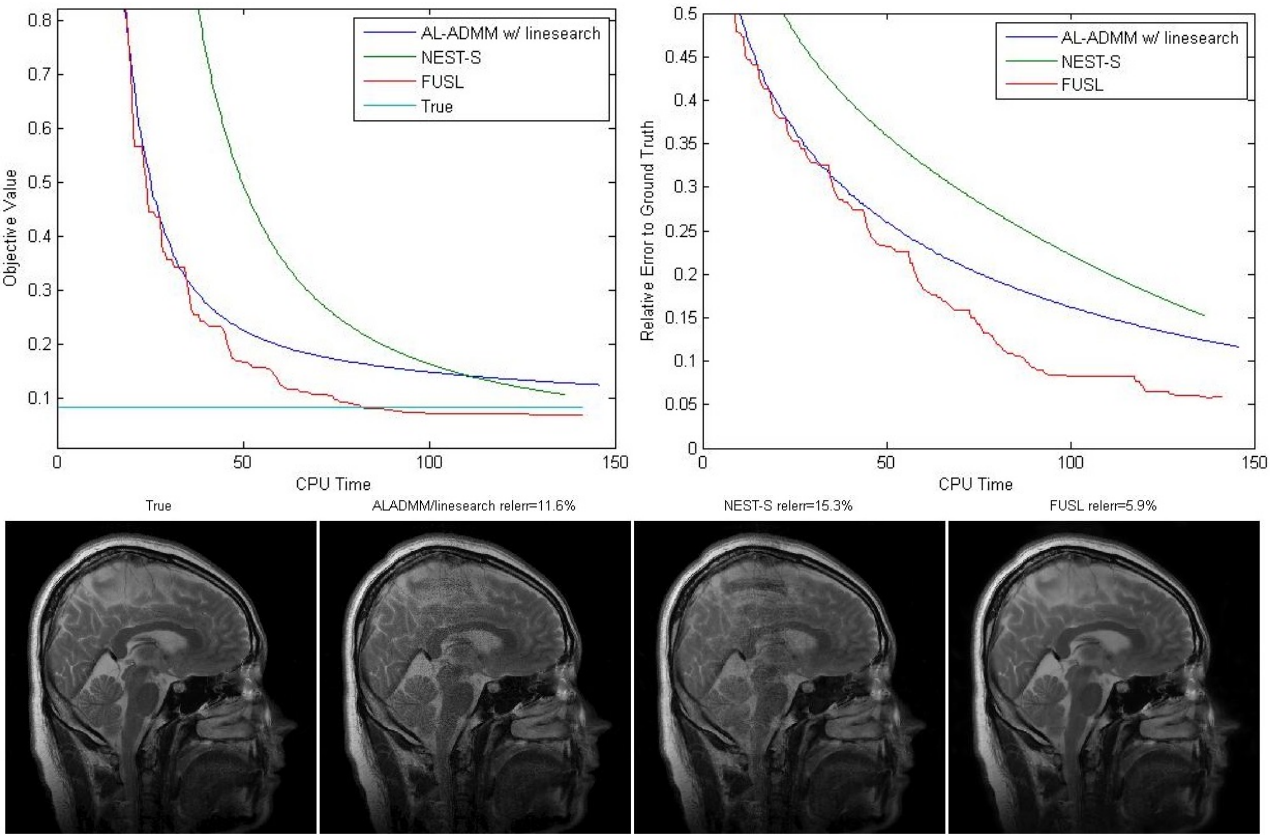

FIG. 5.4. *PPI image reconstruction (acquisition rate:* 14%*)*

[1] Stephen Becker, Jérôme Bobin, and Emmanuel J Candès. Nesta: a fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, 4(1):1–39, 2011.

[2] JY Bello Cruz and W de Oliveira. Level bundle-like algorithms for convex optimization. *Journal of Global Optimization*, pages 1–23, 2013.

[3] A. Ben-Tal and A. S. Nemirovski. Non-Euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102:407–456, 2005.

[4] Ulf Bränlund, Krzysztof C Kiwiel, and Per Olof Lindberg. A descent proximal level bundle method for convex nondifferentiable optimization. *Operations Research Letters*, 17(3):121–126, 1995.

[5] Y. Chen, G. Lan, and Y. Ouyang. Optimal primal-dual methods for a class of saddle point problems. *SIAM Journal on Optimization*, 2014. to appear.

[6] Yunmei Chen, William Hager, Feng Huang, Dzung Phan, Xiaojing Ye, and Wotao Yin. Fast algorithms for image reconstruction with application to partially parallel mr imaging. *SIAM Journal on Imaging Sciences*, 5(1):90–118, 2012.

[7] W de Oliveira and CLAUDIA Sagastizábal. Level bundle methods for oracles with on-demand accuracy. *Optimization Methods and Software*, (ahead-of-print):1–30, 2014.

[8] J.E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.

[9] K.C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46:105–122, 1990.

[10] K.C. Kiwiel. Proximal level bundle method for convex nondifferentable optimization, saddle point problems and variational inequalities. *Mathematical Programming, Series B*, 69:89–109, 1995.

[11] KC Kiwiel. Bundle methods for convex minimization with partially inexact oracles. *Comput. Optim. Appl., to appear*, 2009.

[12] Krzysztof C Kiwiel. A proximal bundle method with approximate subgradient linearizations. *SIAM Journal on optimization*, 16(4):1007–1023, 2006.

[13] Krzysztof C Kiwiel and Claude Lemaréchal. An inexact bundle variant suited to column generation. *Mathematical programming*, 118(1):177–206, 2009.

[14] G. Lan. Bundle-level type methods uniformly optimal for smooth and non-smooth convex optimization. Manuscript, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA, 2013. Mathematical Programming (to appear).

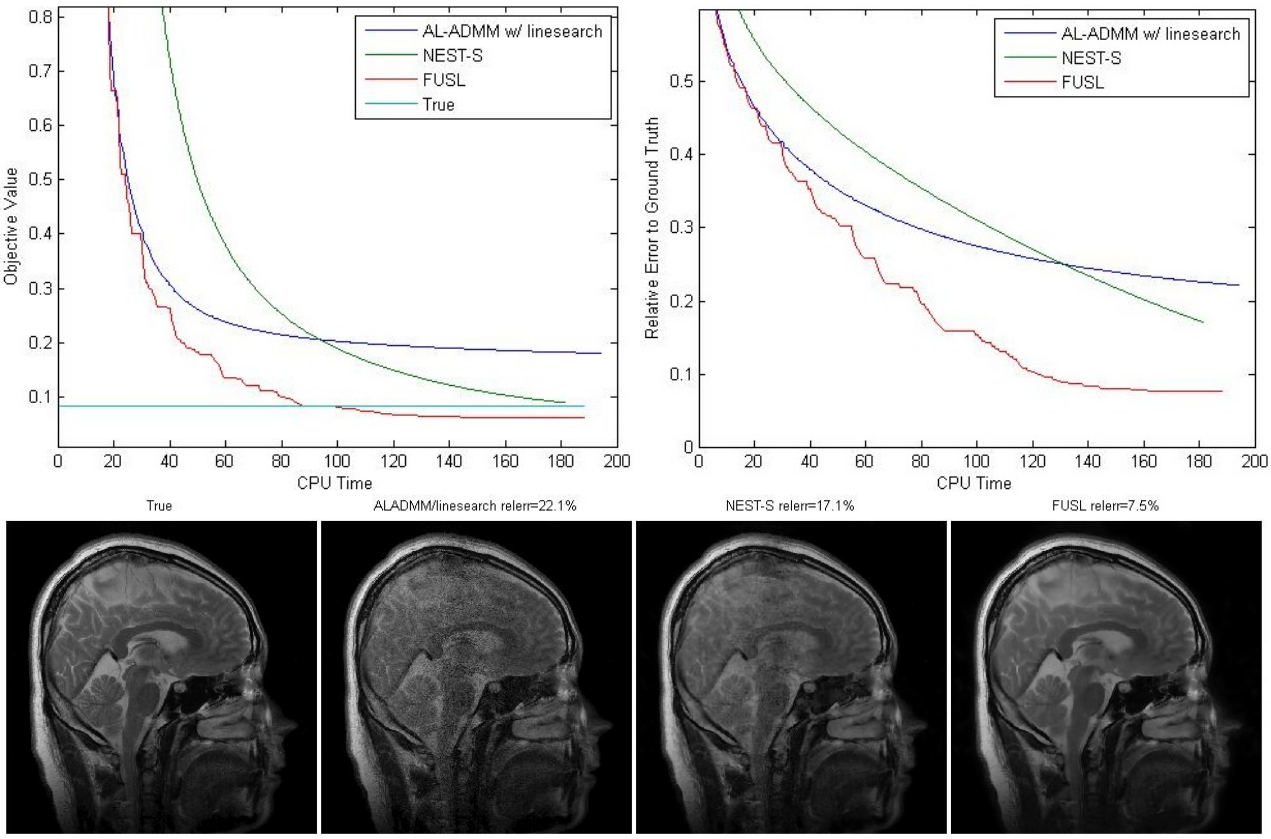

FIG. 5.5. *PPI image reconstruction (acquisition rate:* 10%*)*

[15] G. Lan. Bundle-type methods uniformly optimal for smooth and non-smooth convex optimization. Manuscript, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA, November 2010.

[16] C. Lemaréchal, A. S. Nemirovski, and Y. E. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–148, 1995.

[17] Mosek. The mosek optimization toolbox for matlab manual. version 6.0 (revision 93). `http://www.mosek.com`.

[18] A. S. Nemirovski and D. Yudin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley, XV, 1983.

[19] Y. E. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN SSSR*, 269:543–547, 1983.

[20] Y. E. Nesterov. *Introductory Lectures on Convex Optimization: a basic course*. Kluwer Academic Publishers, Massachusetts, 2004.

[21] Y. E. Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103:127–152, 2005.

[22] W Oliveira, C Sagastizábal, and C Lemaréchal. Bundle methods in depth: a unified analysis for inexact oracles. *Optim Online Rep (Submitted)*, 2013.

[23] Welington Oliveira, Claudia Sagastizábal, and Susana Scheimberg. Inexact bundle methods for two-stage stochastic programming. *SIAM Journal on Optimization*, 21(2):517–544, 2011.

[24] Welington de Oliveira and Claudia Sagastizábal. Bundle methods in the xxist century: A bird's-eye view. *Pesquisa Operacional*, 34(3):647–670, 2014.

[25] Yuyuan Ouyang, Yunmei Chen, Guanghui Lan, and Eduardo Pasiliao Jr. An accelerated linearized alternating direction method of multipliers. *arXiv preprint arXiv:1401.6607*, 2014.

[26] Peter Richtárik. Approximate level method for nonsmooth convex minimization. *Journal of Optimization Theory and Applications*, 152(2):334–350, 2012.

[27] Wim van Ackooij and Claudia Sagastizábal. Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. *SIAM Journal on Optimization*, 24(2):733–765, 2014.