Article

An Adaptive Proximal Bundle Method with Inexact Oracles for a Class of Nonconvex and Nonsmooth Composite Optimization

Xiaoliang Wang *, Liping Pang, Qi Wu and Mingkun Zhang

School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China; lipang@dlut.edu.cn (L.P.); wuqi215010@mail.dlut.edu.cn (Q.W.); mkzhang@mail.dlut.edu.cn (M.Z.)
* Correspondence: xliangwangdlut@mail.dlut.edu.cn

Abstract: In this paper, an adaptive proximal bundle method is proposed for a class of nonconvex and nonsmooth composite problems with inexact information. The composite problems are the sum of a finite convex function with inexact information and a nonconvex function. For the nonconvex function, we design the convexification technique and ensure the linearization errors of its augment function to be nonnegative. Then, the sum of the convex function and the augment function is regarded as an approximate function to the primal problem. For the approximate function, we adopt a disaggregate strategy and regard the sum of cutting plane models of the convex function and the augment function as a cutting plane model for the approximate function. Then, we give the adaptive nonconvex proximal bundle method. Meanwhile, for the convex function with inexact information, we utilize the noise management strategy and update the proximal parameter to reduce the influence of inexact information. The method can obtain an approximate solution. Two polynomial functions and six DC problems are referred to in the numerical experiment. The preliminary numerical results show that our algorithm is effective and reliable.

Keywords: nonconvex and nonsmooth; inexact oracles; disaggregate strategy; proximal bundle method; convexification technology; DC problems

1. Introduction

Consider the following optimization problem:

$$\min_{x \in \mathbb{R}^N} \psi(x) := f(x) + h(x),$$

where $f : \mathbb{R}^N \to \mathbb{R}$ is a finite convex function and function $h$ is not necessarily convex. So the primal function (1) may be nonconvex and note that functions $f$ and $h$ are not necessarily smooth. In this paper, we consider the case that function $h$ is easy to evaluate while the function $f$ is much harder to evaluate and is time consuming.

The sum of two functions can be found in many optimization problems such as the Lasso problem in image problems and the optimization problems in machine learning and so on. Meanwhile, the composite function (1) can be obtained from other problems such as by splitting technique and nonlinear programming and so on. Concretely, in some cases, the function considered is much complicated and difficult to evaluate, to speed up calculations, dividing the primal function into two functions $f$ and $h$ with relatively simple structure is a possible way. Besides that, another way is the penalty strategy which transfers the constraint problem into an unconstrained problem with the sum form.

Note that the splitting type methods (see [1,2]) and the alternating type methods (see [3,4]) are two classes of important methods for composite optimization. When functions $f$ and $h$ have some special structures, the above methods may be effective and own better convergent results. However, if the functions do not own special structures or the functions are much complex and difficult to evaluate, the above methods may not be
suitable for Problem (1). Meanwhile, in the alternating direction type methods, at least two subproblems need to be solved at each iteration, if one of the subproblems is difficult or hard to solve, the algorithms’ effectiveness may be slowed down. Then, it is meaningful to seek other suitable methods to deal with Problem (1) without special structures.

In recent years, many scholars have devoted time to seeking effective methods for nonconvex and nonsmooth optimization problems, see [5–20]. Usually, bundle methods are much effective for solving nonsmooth optimization problems [21–26]. Bundle methods use “black box” to compute the objective value and one of its subgradients (not special) at each iterations. Then, bundle techniques can be a class of possible effective ways to deal with the composite problem (1). At present, the proximal alternating linearization type methods (see [4,27–29]) are one effective kind of bundle methods for some composite problems. They need to solve two subproblems at each iteration and the data referred are usually exact. When inexact oracles are involved, the above methods may not be suitable and even be not convergent.

In this paper, we design a proximal bundle method for the inexact composite problem (1) and update the proximal parameter \( \mu \) to reduce the effects of the inexact information. In the following, we first present some cases where inexact evaluations are generated.

Inexact evaluations are usually referred to in stochastic programming and Lagrangian relaxation [30,31]. It is at the very least impractical and is usually not even easy to solve those subproblems exactly. In bundle methods, inexact information are obtained from inexact oracles. There are different types of inexact oracles. In our work, we consider the Upper oracle (see (2a)–(2c) below). The Upper oracles may overestimate the corresponding function values and get negative linearization errors even if the primal function is convex.

In this paper, we focus on a class of nonconvex and nonsmooth composite problems with inexact data. The design and convergence analysis of bundle methods for nonconvex problems with inexact function and subgradient evaluations are quite involved and there are only a handful of papers for this topic, see [15,32–35].

In this paper, we present a proximal bundle method with a convexification technique and noise management strategy to solve composite problem (1). Concretely, we design “convexification” technique for the nonconvex function \( h \) to make sure the corresponding linearization error nonnegative and then we adopt the noise management strategy for inexact function \( f \). If the error is “too” large and the testing condition (22) is not satisfied, we decrease the value of the proximal parameter \( \mu \) to obtain better iterative point. We summary our work as follows:

- Firstly, we design the convexification technique for the nonconvex function \( h \) to make sure the linearization errors of the augment function \( \phi_n = h + \frac{\eta_n}{2} \| \cdot \|^2 \) are nonnegative. Note that the the augment function \( \phi_n \) may not be a convex function, the nonnegative linearization errors can be obtained by the choices of the parameter \( \eta_n \). Similar strategy can also be seen in [10,11,15,16].
- Then, the sum of functions \( f \) and \( \phi_n \) is regarded as an approximate function for the composite function (1). We construct respectively the cutting plane models for functions \( f \) and \( \phi_n \) and regard the sum of the two cutting plane models as the cutting plane model for the approximate function which may be a better cutting plane model. It should be noted that since inexact information are referred, the corresponding cutting plane model may not always be below function \( f \).
- Although we design the cutting plane models for functions \( f \) and \( \phi_n \), respectively, only one quadratic programming (QP) subproblem needs to be solved at each iteration. By the construction of cutting plane models, we have that the QP subproblem is strictly convex and has unique solution which makes our algorithm more effective.
- In the method, we construct the noise management step to deal with the inexact data for function values and subgradient values where the errors are only bounded and need not to vanish. If the noise error is “too” large and the testing condition (22) is not satisfied, we decrease the value of \( \mu \) to obtain a better iterative point.
Two polynomial functions with twenty different dimensions and six DC (difference of convex) problems are referred to in the numerical experiment. In exact cases, our method is comparable with the method in [16] and has higher precision. In five different types of inexact oracles, we obtain that the exact case has the best performance and the performance of the vanishing error cases are generally better than that in the constant error cases. We also apply our method to six DC problems and the results show that our algorithm is effective and reliable.

The remainder of this paper is organized as follows. In Section 2, we review some variational analysis definitions and some preliminaries for proximal bundle method. Our proximal bundle method is given in Section 3. In Section 4, we present the convergent property of the algorithm. Some preliminary numerical testings are implemented in Section 5. In Section 6, we give some conclusions.

2. Preliminaries

In this section, we firstly review some concepts and definitions and then present some preliminaries for a proximal bundle method.

2.1. Preliminary

In this subsection, we recall concepts and results of variational analysis that will be used in the latter of the paper. The definition of lower-C\textsuperscript{k} is given in Definition 10.29 in [36]. For completeness, we state it as follows:

\textbf{Definition 1.} A function \( F : \mathcal{O} \to \mathbb{R} \), where \( \mathcal{O} \) is an open subset of \( \mathbb{R}^N \), is said to be lower-C\textsuperscript{k} on \( \mathcal{O} \), if on some neighborhood \( V \) of each \( \hat{x} \in \mathcal{O} \), there is a representation:

\[ F(x) = \max_{t \in T} F_t(x), \]

in which the function \( F_t \) are of class \( C^k \) on \( V \) and the index set \( T \) is a compact space such that \( F_t(x) \) and all its partial derivatives through order \( k \) depend continuously not just on \( x \) but on \( (t, x) \in T \times V \).

If \( k = 2 \), \( F \) is a lower-C\textsuperscript{2} function. lower-C\textsuperscript{2} function has a special relationship with convexity, see Theorem 10.33 in [36]. We state its equivalent statement as follow: A function \( F \) is lower-C\textsuperscript{2} on an open set \( \mathcal{O} \subseteq \mathbb{R}^N \) if \( F \) is finite on \( \mathcal{O} \) and for any \( x \in \mathcal{O} \), there exists a threshold \( \lambda \geq 0 \) such that \( F + \frac{\lambda}{2} \| \cdot \|^2 \) is convex on an open neighborhood \( V \) of \( x \) for all \( \lambda \geq \lambda \). Specifically, if the function \( F \) is convex and finite-valued, then \( F \) is lower-C\textsuperscript{2} with threshold \( \lambda = 0 \).

For nonconvex function \( h \), in the following, we consider that \( h \) is a lower-C\textsuperscript{2} function. Since functions \( f \) and \( h \) are all not necessarily smooth, then the composite function (1) is also not necessarily smooth. For proper convex function \( f \), the common subdifferential in convex analysis is used, which is denoted by \( \partial f(x) \) at the point \( x \in \mathbb{R}^N \) (see in [37]). For proper and regular function \( h \), we utilize the limiting subdifferential and also denote it by \( \partial h(x) \) at point \( x \) (see in [36]). The definition of the limiting subdifferential is as follows:

\[ \partial h(x) := \limsup_{y \to x, h(y) \to h(x)} \left\{ v \mid \liminf_{u \to y, u \neq y} \frac{h(u) - h(y) - \langle v, u - y \rangle}{\|u - y\|} \geq 0 \right\}. \]

In nonsmooth analysis for convex function \( f, \varepsilon - \) subdifferential at point \( x^k \) is usually used, which is defined:

\[ \partial_\varepsilon f(x^k) := \left\{ g \mid f(x) \geq f(x^k) - \varepsilon + \langle g, x - x^k \rangle \right\}, \]

where \( \varepsilon \geq 0 \). In the following, we present the inexact data for function \( f \) and give some preliminaries for proximal bundle method.
2.2. Inexact Information and Bundle Construction

Bundle methods are much effective for nonsmooth problems and always utilize “black box” to compute function value and one subgradient at each iterative point. It should be noted that the obtained subgradient is not special. Along the iterative progress, the generated points are divided into two styles: null points, used essentially to increase the model’s accuracy; serious points, significantly decreasing the objective function (and also improve the approximate model’s accuracy). The corresponding iterations are respectively called null steps and serious steps. In the literature, serious points are sometimes called as prox-center or stability center, denoted by $\bar{x}^{k(n)}$. Then, the sequence $\{\bar{x}^{k(n)}\}$ is a subsequence of the sequence $\{x^n\}$. For notation simplification, we write $\bar{x}^k = \bar{x}^{k(n)}$.

For function $f$, the oracle can only provide inexact function value and one subgradient value at each iteration, $f^l := f_{x^l}, g^l_f := g^l_{x^l}$, with unknown but bounded inaccuracy. That is, for $\forall x^l \in \mathbb{R}^N$, we have
\[
\begin{align*}
    f^l & \geq f(x^l) - \theta_l, \quad (2a) \\
    f(\cdot) & \geq f^l + \langle g^l_f, \cdot - x^l \rangle - \epsilon_l, \quad (2b)
\end{align*}
\]

Meanwhile
\[
\theta_l \leq \bar{\theta}_l, \quad \text{and} \quad \epsilon_l \leq \bar{\varepsilon}.
\]

According to (2a)–(2c), we have the following relationships
\[
\begin{align*}
    g^l_f & \in \partial_{\theta_l + \epsilon_l} f(x^l), \\
    f^l & \in [f(x^l) - \theta_l, f(x^l) + \epsilon_l].
\end{align*}
\]

Note that we only require the relationship $\theta_l + \epsilon_l \geq 0$ holds for each index $l$. The bundle for function $f$ can be noted as
\[
\mathcal{B}_l^f := \left\{ (x^l, f^l = f_{x^l}, g^l_f \in \partial f_{x^l}) \mid l \in I_n \right\},
\]

Now we present the cutting plane model of function $f$ by the inexact information:
\[
\tilde{\varphi}_n(x) = \max_{l \in I_n} \left\{ f^l + \langle g^l_f, x - x^l \rangle \right\} = \bar{f}^k + \max_{l \in I_n} \left\{ -\epsilon_{f,l} + \langle g^l_f, x - \bar{x}^k \rangle \right\},
\]

where $\bar{f}^k = f_{x^{k(n)}}, \bar{x}^k = x^{k(n)}$ is the current stability center with index $k(n)$ corresponding to its candidate point index and $\epsilon_{f,l}$ is linearization errors, which measures the difference between cutting planes and the function value computed in the oracle for the current serious point, that is
\[
\epsilon_{f,l} = f^k - f^l - \langle g^l_f, x^k - x^l \rangle.
\]

Especially, note that the relation $\tilde{\varphi}_n(x) \leq f(x)$ does not necessarily hold. So the linearization error $\epsilon_{f,l}$ may be negative. In fact, by (2a), (2b) and (6), $\epsilon_{f,l}$ satisfies that
\[
\epsilon_{f,l} \geq - (\theta_{k(n)} + \epsilon_l).
\]

Meanwhile, cutting plane model $\tilde{\varphi}_n$ may overestimate $f$ at some points. By (2b), the following inequality holds
\[
\tilde{\varphi}_n(x) \leq f(x) + \max_{l \in I_n} \epsilon_l.
\]

For nonconvex function $h$, linearization errors may be nonnegative. In bundle methods, nonnegative linearization errors are much important for the convergence. For that we present a local “convexification” technique, similar techniques can also be seen in [15,16,38]. For the convexification parameter $\eta_{ln}$, its choice is as follows
\[
\eta_n \geq \eta_n^{\text{min}} = \max \left\{ -\frac{c_{h,l}^k}{\frac{1}{2}\|x^l - \hat{x}^k\|^2} \right\}, \quad \text{with } l \in I_n, \text{ and } \|x^l - \hat{x}^k\|^2 \neq 0,
\]

where \(I_n\) denotes an index set, i.e., \(I_n \subseteq \{0, 1, 2, \ldots\}\) and \(c_{h,l}^k\) is the linearization error of \(h\), which is defined as follows with \(h^l = h(x^l), g_h^l \in \partial h(x^l)\)
\[
e\]

\[
e
\]

\[
\hat{x}_{h,l} = h(\hat{x}^k) - h^l - \langle g_h^l, \hat{x}^k - x^l \rangle.
\]

The bundle for function \(h\) can be noted as
\[
B_h^k := \left\{ (x^l, h^l = h(x^l), g_h^l \in \partial h(x^l)), l \in I_n \right\},
\]

Next, we introduce the augment function \(\phi_n\) of \(f\), it is defined by
\[
\phi_n(x) := h(x) + \frac{\eta_n}{2} \|x - \hat{x}^k\|^2, \quad \forall x \in \mathbb{R}^N,
\]

where \(\eta_n \geq \eta_n^{\text{min}}\) holds. Note that by the definition of \(\phi_n\), we have \(h(\hat{x}^k) = \phi_n(\hat{x}^k)\). By the calculation of subgradient, we have there exists \(g_h^l \in \partial h(x^l)\) satisfying
\[
g_h^l = g_h^l + \eta_n (x^l - \hat{x}^k) \in \partial \phi_n(x^l).
\]

Meanwhile, the linearization error of function \(\phi_n\) is
\[
e_{\phi,l}^k = \phi_n(\hat{x}^k) - \phi_n(x^l) - \langle g_h^l, \hat{x}^k - x^l \rangle
\]
\[
= h(\hat{x}^k) - h(x^l) - \frac{\eta_n}{2} \|x^l - \hat{x}^k\|^2 - \langle g_h^l + \eta_n (x^l - \hat{x}^k), \hat{x}^k - x^l \rangle = e_{h,l}^k + \frac{\eta_n}{2} \|x^l - \hat{x}^k\|^2.
\]

By the choice of the convexification parameter \(\eta_n\), we have \(e_{\phi,l}^k \geq 0\) holds for all \(l \in I_n\).

In the following, we regard the sum of functions \(f\) and \(\phi_n\) as an approximate function for composite function (1):
\[
\Psi_n(x) = f(x) + \phi_n(x).
\]

For (13), we utilize the sum of cutting plane models of functions \(f\) and \(\phi_n\) as the cutting plane model. The cutting plane model of the augment function \(\phi_n\) is defined as follows
\[
\tilde{\phi}_n(x) = \max_{l \in I_n} \left\{ \phi_n(x^l) + \langle g_h^l, x - x^l \rangle \right\}.
\]

Its equivalent form is
\[
\tilde{\phi}_n(x) = \phi_n(\hat{x}^k) + \max_{l \in I_n} \left\{ -e_{\phi,l}^k + \langle g_h^l, x - \hat{x}^k \rangle \right\}
\]
\[
= h(\hat{x}^k) + \max_{l \in I_n} \left\{ -e_{h,l}^k + \frac{\eta_n}{2} \|x^l - \hat{x}^k\|^2 + \langle g_h^l + \eta_n (x^l - \hat{x}^k), x - \hat{x}^k \rangle \right\}.
\]

Then, the cutting plane model for the approximate function \(\Psi_n\) is
\[
\Phi_n(x) = \tilde{\phi}_n(x) + \tilde{\phi}_n(x).
\]

The new iterative point \(x^{n+1}\) is given by the following QP (quadratic programming) subproblem
\[
x^{n+1} := \arg \min_{x \in \mathbb{R}^N} \left\{ \tilde{\phi}_n(x) + \tilde{\phi}_n(x) + \frac{\mu_n}{2} \|x - \hat{x}^k\|^2 \right\} = \arg \min_{x \in \mathbb{R}^N} \left\{ \Phi_n(x) + \frac{\mu_n}{2} \|x - \hat{x}^k\|^2 \right\},
\]

where \(\mu_n > 0\) is the proximal parameter. Note that \(x^{n+1}\) is the unique solution to (15) by strong convexity. The following lemma shows the relation between the current stationary
center and the new generated point. Similar conclusion can also be seen in Lemma 10.8 in [39] which is for convex function. Here we omit the proof.

Lemma 1. Let \( x^{n+1} \) be the unique solution to the QP subproblem (15) and proximal parameter \( \mu_n > 0 \). Then, we have

\[
\chi^{n+1} = \bar{x}^k - \frac{1}{\mu_n} G^n,
\]

where

\[
G^n := \sum_{i \in I_n} a_{1i} \hat{g}_j + \sum_{i \in I_n} a_{2i} \left( \hat{s}_{ih} + \eta_n (x^l - \bar{x}^k) \right).
\]

Meanwhile \( a_1 = (a_{11}, \ldots, a_{1n}) \) and \( a_2 = (a_{21}, \ldots, a_{2n}) \) is the solution to

\[
\begin{align*}
\min_{a_1, a_2 \in \mathbb{R}^{|I_n|}} & \quad \frac{1}{2\mu_n} \| G^n \|^2 + \sum_{i \in I_n} a_{1i}^2 e_f^k + \sum_{i \in I_n} a_{2i}^2 \phi_f^k \quad \text{s.t.} \quad a_1, a_2 \in S_n := \left\{ z \in [0, 1]|^{|I_n|} : \sum_{i \in I_n} z_i = 1 \right\}.
\end{align*}
\]

In addition, the following relations hold:

(i) \( G^n \in \partial \Phi_n (x^{n+1}) \);

(ii) \( f^k + \hat{\phi}_n (\hat{x}^k) - \Phi_n (x^{n+1}) = \mu_n \| x^{n+1} - \bar{x}^k \|^2 + \sum_{i \in I_n} a_{1i}^l e_f^k + \sum_{i \in I_n} a_{2i}^l \phi_f^k \).

In the following, we present the concepts of the predict descent. Concretely, the predict descent for functions \( f, \phi_n, \Psi_n \) are stated as follows

\[
\begin{align*}
\delta_{\mu+1}^f &= \hat{f}^k - \tilde{\phi}_n (x^{n+1}), \\
\delta_{\mu+1}^h &= h(\hat{x}^k) + \frac{\eta_n}{2} \| x^{n+1} - \bar{x}^k \|^2 - \tilde{\phi}_n (x^{n+1}), \\
\delta_{\mu+1}^\mu &= \tilde{\phi}_{\mu+1} + \delta_{\mu+1}^h.
\end{align*}
\]

Note that the predict descent is much important for the convergence of bundle methods. By the definitions of functions \( \phi_n \) and \( \tilde{\phi}_n \), we have \( \delta_{\mu+1}^\mu \geq 0 \). Since inexact data are referred to in the computation of function \( f \), the nonnegativity of \( \delta_{\mu+1}^f \) and \( \delta_{\mu+1}^\mu \) cannot be guaranteed. Hence the nonnegativity of \( \delta_{\mu+1}^h \) cannot be guaranteed too.

Next we give the aggregate linearization error which is defined by

\[
e_{\mu+1} := \sum_{i \in I_n} a_{1i}^l e_f^k + \sum_{i \in I_n} a_{2i}^l \phi_f^k.
\]

By the term (ii) in Lemma 1 and the definition of \( \delta_{\mu+1} \) in (18), the following relationship holds

\[
\delta_{\mu+1} = \| G^n \|^2 + \frac{\eta_n}{2} \| x^{n+1} - \bar{x}^k \|^2 + e_{\mu+1}
\]

\[
= \frac{R_n + \mu_n}{2} \| x^{n+1} - \bar{x}^k \|^2 + e_{\mu+1},
\]

where \( R_n = \mu_n + \eta_n \). We define the aggregate linearization for approximate model \( \Phi_n \):

\[
\Phi_n^{lin} (x) := \Phi_n (x^{n+1}) + \langle G^n, x - x^{n+1} \rangle.
\]

Then, the aggregate linearization error can be also expressed as the difference between the value of the oracle at the current serious point and the value of aggregate linearization \( \Phi_n^{lin} \) at that point, that is,

\[
e_{\mu+1} = f^k + h(\hat{x}^k) - \Phi_n^{lin} (\hat{x}^k).
\]
Indeed, by the definition of $\Phi_{ln}(x)$, we have

$$\Phi_{ln}(x) = f^k + h(x^k) + \langle G^n, x - x^k \rangle - e_{n+1}.$$ 

By the convexity of function $\Phi_n$, the inequality $\Phi_n(x) \geq \Phi_{ln}(x)$ holds. So for any $x \in \mathbb{R}^N$, we have

$$f^k + h(x^k) \leq \Phi_n(x) - \langle G^n, x - x^k \rangle + e_{n+1}.$$ 

By (8), the following inequality holds under the condition $\tilde{\Phi}_n(x) \leq \phi_n(x)$:

$$f^k + h(x^k) \leq \psi(x) + \frac{\eta_n}{2} \| x - \hat{x}^k \|^2 + \max_{l \in I_n} \| \tilde{l} \| - \langle G^n, x - x^k \rangle + e_{n+1}. \tag{21}$$

Note that the condition $\tilde{\Phi}_n(x) \leq \phi_n(x)$ may not be hold if the convexification parameter $\eta_n$ is less that the threshold parameter $\bar{\eta}$ (the function $\phi_n(x)$ may not be convex), but the choice of $\eta_n$ ensures the nonnegativity of $e^k_{\phi}$ for all $l \in I_n$.

By the nonnegativity of $e^k_{\phi}$ and (7), the aggregate linearization error satisfies

$$e_{n+1} \geq - \left( \theta_{h(n)} + \sum_{l \in I_n} a_l^1 \epsilon_l \right) \geq - (\bar{\theta} + \bar{\epsilon}).$$

Using the fact that $x^{n+1}$ is the solution of the QP problem (15) and the definition of predict descent in (18), we have that

$$\delta_{n+1} \geq \frac{\mu_n}{2} \| x^{n+1} - \hat{x}^k \|^2 - \left( \tilde{\phi}_n(x^k) + \tilde{\Phi}_n(x^k) - f^k - \phi_n(x^k) - \frac{\eta_n}{2} \| x^{n+1} - x^k \|^2 \right)$$

$$\geq \frac{\mu_n}{2} \| x^{n+1} - \hat{x}^k \|^2 + \frac{\eta_n}{2} \| x^{n+1} - x^k \|^2 - \left( \tilde{\phi}_n(x^k) - f^k \right)$$

$$= \frac{R_n}{2} \| x^{n+1} - \hat{x}^k \|^2 - \left( \tilde{\phi}_n(x^k) - f^k \right),$$

where the second inequality follows from the nonnegativity of $e^k_{\phi}$. By (5) and $x = \hat{x}^k$, we have $\tilde{\phi}_n(x^k) - f^k = \max_{l \in B_{\phi}(\{ -e^k_{\phi} \})} \left\{ -e^k_{\phi} \right\}$ holds, Note that if only “small” errors have been introduced into the model $\Phi_n$, then it holds

$$\delta_{n+1} \geq \frac{\eta_n}{2} \| x^{n+1} - x^k \|^2 + \frac{\mu_n}{2} \| x^{n+1} - x^k \|^2 = \frac{R_n}{2} \| x^{n+1} - \hat{x}^k \|^2. \tag{22}$$

Then, by (20) and (16), (22) has the following equivalent forms:

$$\begin{cases} (22) \text{ holds, if and only if } \frac{\| G^n \|^2}{2\mu_n} > -e_{n+1}, \\ (22) \text{ holds, if and only if } \delta_{n+1} > \frac{\| G^n \|^2}{2\mu_n} + \frac{\eta_n}{2} \| x^{n+1} - \hat{x}^k \|^2. \tag{23} \end{cases}$$

Next, we present a optimality measure. Concretely, it is that

$$V_n := \max \{ \| G^n \|, e_{n+1} \}.$$ 

By the above discussions, we have

$$V_n \leq \max \left\{ \sqrt{2\mu_n\delta_{n+1}}, \delta_{n+1} \right\}, \quad \text{if (22) holds},$$

$$V_n \leq \sqrt{2\mu_ne_{n+1}} \leq \sqrt{2\mu_n(\bar{\theta} + \bar{\epsilon})}, \quad \text{otherwise.} \tag{24}$$

From the above inequalities, the smaller $\mu_n$ may lead to higher probability to make inequality (22) hold. Based on that, we will update the parameter $\mu_n$ to reduce the effects of errors.
In the next section, we will give our proximal bundle algorithm for the primal composite problem \((1)\) with inexact information.

3. Algorithm

In this section, we present our adaptive bundle algorithm to composite problem \((1)\) with inexact information. To handle inexact information, similar to [17], we introduce the noise management step. Concretely, when the condition \((22)\) does not hold, \(\mu_n\) is reduced in order to make \(\delta_{n+2} > \delta_{n+1}\) and increase the probability of condition \((22)\).

| Algorithm 1 (Nonconvex Nonsmooth Adaptive Proximal Bundle Method with Inexact Information for a class of composite optimization) |
|---|
| **Step 0 (Input and Initialization):** |
| Choose initial point \(x^0 \in \mathbb{R}^N\), constants \(\kappa, m_1 \in (0,1)\), an unacceptable increase parameter \(M_0 > 0, \mu_{\text{max}} > 0, R_0 > 0, \tau > 1, \gamma \geq 1\) and a stopping tolerance \(\text{Tol} \geq 0\). Set noise management parameter \(\text{NMP} = 0\) and \(\hat{x}^0 = x^0\). Call the black box to compute \(f_{\hat{g}_0}, \gamma_{\hat{g}_0}, h(\hat{x}^0), \beta_{\hat{g}_0}\). Set \(n = 0\). |

| **Step 1 (Model generation and QP subproblem):** |
| Having the current proximal center \(\hat{x}^k\), the current bundles \(B^f_I\) and \(B^g_{\mu,1}\) with index set \(I_{\mu}\), and the current proximal parameter \(\mu_n\) and the convexification parameter \(\eta_n\). Having the current approximate models \(\hat{\phi}_n(x)\) and \(\hat{\phi}_n(x)\). Compute the QP problem \((15)\) to get the next iterative point \(x^{n+1}\) and simplex multipliers \((\sigma_1, \sigma_2)\). Then, compute \(G^n, \delta_{n+1}, e_{n+1}\) and \(V_n\). |

| **Step 2 (Stopping criterion):** |
| If \(V_n \leq \text{Tol}\), then stop. Otherwise, go to Step 3. |

| **Step 3 (Noise Management):** |
| If relationship \((22)\) does not hold, set \(\text{NMP} = 1, \mu_{n+1} = \kappa \mu_n, n := n + 1\), go to Step 1; otherwise, set \(\text{NMP} = 0\) and call the noise is acceptable and go to Step 4. |

| **Step 4 (Descent testing):** |
| Call the black box to compute \((\hat{f}^{n+1}, \hat{g}^{n+1})\) and \((h^{n+1}, g_{h}^{n+1})\). Check the descent condition \[\hat{f}^k + h(\hat{x}^k) - \hat{f}^{n+1} - h^{n+1} - \frac{\eta_n}{2} ||x^{n+1} - \hat{x}^k||^2 \geq \gamma_1 \delta_{n+1},\] (25) if \((25)\) does not hold, then declare a null step, and set \(k(n + 1) = k(n)\). If \(\text{NMP} = 0\), choose \(\mu_{n+1} \in [\gamma \mu_n, \mu_{\text{max}}]\); else if \(\text{NMP} = 1\), take \(\mu_{n+1} = \mu_n\). Update the bundle information and go to Step 5. Otherwise, declare a serious step. Set \(x^{k+1} = x^{n+1}\), \(\text{NMP} = 0, f_{\hat{x}^{k+1}} = f_{x^{k+1}}, k(n + 1) = k + 1\) and choose \(\mu_{n+1} = \mu_n\), update the bundle information and go to Step 5. |

| **Step 5 (Update parameter):** |
| Apply the rule to compute \(\eta_{n+1}\) |
| \[\left\{ \begin{array}{l} \eta_{n+1} = \eta_n, \quad \text{if } \eta_n \geq \eta_{n+1}^{\min}, \\ \eta_{n+1} = \tau \eta_{n+1}^{\min}, \quad \text{if } \eta_n < \eta_{n+1}^{\min} \end{array} \right.\] (26) |

where \(\eta_{n+1}^{\min}\) is given by \((9)\), written with \(n\) replaced by \(n + 1\). |

| **Step 6 (Restart step):** |
| If \(\psi(x^{n+1}) > \psi(\hat{x}^k) + M_0\) holds, then the objective increase is unacceptable; Restart the algorithm by setting \(\eta_0 := \eta_n, \mu_0 := \tau \mu_n, R_0 := \eta_0 + \mu_0, x^0 := \hat{x}^k, k(0) := 0, i_0 := 0, I_0 := \{0\}, n := 0\), where \(i_0\) is the index of serious points. Then, loop to Step 1; otherwise, increase \(k\) by 1 in the case of the serious step. In all cases, increase \(n\) by 1, and loop to Step 1. |

**Remark 1.** Note that in Algorithm 1, the update of elements in bundles are not stated clearly. For null step and serious step, the updating strategies are different. When a serious step occurs, the new generated point is regarded as a new proximal center and the corresponding linearization errors in the bundles all should be updated. When a null step emerges, the proximal center keeps
unchanged and only the new generated information are added into the bundles to improve the model’s accuracy. As the iterations proceed, the elements in the bundles may be too large that reduces the efficiency of the algorithm. Then, the active technology (only the active element $a_1$ and $a_2$ are kept in the bundles) and the compression strategy can be adopted. For the compression strategy, the number of elements in the bundles can be at least two, the aggregate information and the new generated information. It should be noted that although the compression strategy does not impair the convergence of the algorithm, it may affect the model’s effectiveness if the number of elements in the bundles is too small.

In the following, we will focus on the analysis of Algorithm 1 which indicates the algorithm is well defined. If the algorithm loops forever, three situations may occur (the number of restart steps are finite, which can be see in Lemma 3):

- a very large number of serious steps.
- a finite number of serious steps, followed by an very large number of null steps;
- a very large loop of noise management.

Next, we firstly give the case of very large loop of noise management.

**Lemma 2.** If an very large loop between Step 1 and Step 3 in Algorithm 1 occurs, then the optimal measure $V_n \to 0$.

**Proof.** Suppose an very large loop between Step 1 and Step 3 begins at iterative index $\bar{l}$. According to the algorithm, this means that for all $n \geq \bar{l}$, neither the proximal center $\hat{x}^{(n)} = \hat{x}^{(\bar{l})}$ nor the approximate models $\hat{g}_n, \hat{h}_n, \Phi_0$ change. Hence, when solving sequentially the QP optimization problem (15), only parameter $\mu_n$ is updated. By the strategy for $\mu_n$, we have $\mu_n = \kappa^{n-1} \mu_\ell$, then it holds $\mu_n \to 0$ as $n \to \infty$ by $\kappa \in (0, 1)$. Using (24), we have

$$0 \leq V_n \leq \sqrt{2\mu_n(\bar{\theta} + \bar{\epsilon})} \to 0, \quad \mu_n \to 0 \text{ when } n \to \infty.$$  

Then, the proof is completed. □

Note that if very large noise management steps happen, there is finite number of update for the convexification parameter $\eta_n$. Then, $\eta_n$ eventually is bounded. Before the last two cases, we show that there is only finite number of restart steps in Algorithm 1. For that, we make an assumption, which can also be founded in [16].

**Assumption 1.** The following level set $T := \{x \in \mathbb{R}^N : \psi(x) \leq \psi(x^0) + M_0\}$ is nonempty and compact.

By the definition of lower-C$^2$, the compactness of set $T$ and the finite covering theorem, there exists a threshold $\bar{\rho}$ such that for all $\eta \geq \bar{\rho}$, the augmented function $h(x) + \eta \|x - \hat{x}^k\|^2/2$ is convex for $\forall x, \hat{x}^k \in T$.

The compactness of $T$ allows us to find Lipschitz constants for functions $f$ and $h$, named $L_f$ and $L_h$ respectively (by lower-C$^2$ function’s local lipschitz property and the finite covering theorem). The following lemma indicates that the restart step in Algorithm 1 is finite.

**Lemma 3.** Suppose only finite number of the noise management steps occur, Assumption 1 holds and consider the sequence of iterative points $\{x^n\}$ generated by Algorithm 1. The index $l_k \in I_n$ denotes the current proximal center index, then there can be only a finite number of restart steps in Algorithm 1. Hence, eventually the sequence $\{\hat{x}^k\}$ is entirely in $T$.

**Proof.** Firstly, new iterative point $x^{n+1}$ is well defined by the strong convexity of QP subproblems (15). As functions $f$ and $h$ are lipschitz continuous on the level set $T$ and their lipschitz constants are respective $L_f$ and $L_h$, $\psi(x)$ is also lipschitz continuous in the
compact set $\mathcal{T}$ and one of its lipschitz constants is $L := L_f + L_h$. By the lipschitz continuity of $\psi$, there exists $\epsilon > 0$ such that for any $x \in \{x : \psi(x) \leq \psi(x^0)\}$, the open set $B_{\epsilon}(x)$ is contained in compact set $\mathcal{T}$ (indeed, the choice of $\epsilon = M_0/L$ suffices). Note that:

$$x^{n+1} = p_{\mu_n}(\hat{\phi}_n + \tilde{\phi}_n)(\hat{x}^k) = \arg\min_x \left\{ (\hat{\phi}_n + \tilde{\phi}_n)(x) + \frac{\mu_n}{2}\|x - \hat{x}^k\|^2 \right\}$$

$$\subseteq \left\{ x : (\hat{\phi}_n + \tilde{\phi}_n)(x) + \frac{\mu_n}{2}\|x - \hat{x}^k\|^2 \leq (\hat{\phi}_n + \tilde{\phi}_n)(\hat{x}^k) \right\}$$

$$\subseteq \left\{ x : (\hat{\phi}_n + \tilde{\phi}_n)(\hat{x}^k) + \langle g^k, x - \hat{x}^k \rangle + \frac{\mu_n}{2}\|x - \hat{x}^k\|^2 \leq (\hat{\phi}_n + \tilde{\phi}_n)(\hat{x}^k) \right\}$$

$$\subseteq \left\{ x : \|g^k\|\|x - \hat{x}^k\| + \frac{\mu_n}{2}\|x - \hat{x}^k\|^2 \leq 0 \right\} \subseteq \left\{ x : \|x - \hat{x}^k\| \leq \frac{2L}{\mu_n} \right\},$$

where $l_k \in I_\eta$, and $g^k \in \partial(\hat{\phi}_n + \tilde{\phi}_n)(\hat{x}^k)$. It also holds $g^k \in \partial \psi(\hat{x}^k)$, then $\|g^k\| \leq L$. In Algorithm 1, $\mu_n$ increases when the restart steps and the null steps with $NMP = 0$ happen, eventually proximal parameter $\mu_n$ becomes large enough that the relationship $\frac{2\mu_n}{L} < \epsilon$ holds. Noting that $\psi(\hat{x}^k) \leq \psi(x^0)$ for any new serious point $\hat{x}^k$ generated in Algorithm 1 completes the proof.

Next, we focus on the update of convexification parameter $\eta_n$. The following lemma shows $\eta_n$ eventually keeps unchanged.

**Lemma 4.** Suppose there is a finite number of the noise management steps and Assumption 1 holds. Then, there exists an iteration index $n$ such that for all $n \geq n$, the convexification parameter $\eta_n$ stabilizes, i.e., $\eta_n = \bar{\eta}$. Moreover, if $\bar{\eta} \geq \bar{\rho}$ holds, then for all $n \geq n$, the augmented function $\phi_n(x) = h(x) + \frac{\mu_n}{2}\|x - \hat{x}^k\|^2$ is convex on the compact set $\mathcal{T}$.

**Proof.** By the update of the convexification parameter $\eta_n$ in Algorithm 1, we have $\eta_n$ is nondecreasing: either $\eta_{n+1} = \eta_n$ or $\eta_{n+1} = \tau\eta_n > \eta_n$. Suppose the sequence $\{\eta_n\}$ does not stabilize, there must exist an very large number indices such that the convexification parameter is increased by a factor of at least $\tau$, but that is difficult and can lead to a contradiction. Since there exists an index $\bar{n}$ such that $\eta_{\bar{n}} \geq \bar{\rho}$ and $h(x) + \frac{\mu_n}{2}\|x - \hat{x}^k(n)\|^2$ is convex on the compact set $\mathcal{T}$. For this iteration, we have $\hat{x}^{k}_{\bar{n}+1} + \eta_{\bar{n}}\|x - \hat{x}^k(n)\|^2/2 \geq 0$ for all $l \in I_\eta$ (the linearization error for a convex is always nonnegative). Hence, $\eta_{\bar{n}} \geq \max_{l \in I_\eta} \left\{ \frac{-\hat{x}^{k}_{\bar{n}+1}}{\frac{\mu_n}{2}\|x - \hat{x}^k(n)\|^2} \right\} = \eta_{\bar{n}+1}^{\min}$ holds. Then, from the iteration onward, the convexification parameter will keep unchanged, i.e., $\eta_{n+i} = \eta_{\bar{n}}$ for all $i \geq 1$. Then, the sequence $\{\eta_n\}$ stabilizes. Specially, we choose $\eta = \eta_{\bar{n}}$. For $n \geq n$, if $\eta_n = \eta \geq \bar{\rho}$ holds, then the augmented function $\phi_n(x)$ in $\mathcal{T}$ is convex.

The optimality measure in Algorithm 1 for inexact information is different with that in exact case. The following lemma justifies the choice of $V_n$ as optimality measure and indicates the accumulate point is an approximate solution of primal problem (1).

**Lemma 5.** Suppose there is a finite number of the noise management steps and Assumption 1 holds. Suppose that for an very large subset of iterations $I \subseteq \{0, 1, \cdots\}$, the sequence $\{V_\lambda\}_{\lambda \in I} \rightarrow 0$ as $\lambda \rightarrow \infty$. Let $\{\hat{x}^k(\lambda)\}_{\lambda \in I}$ be the corresponding subsequence of serious points and let $\hat{x}^{acc}$ be an accumulation point. If $\bar{\eta} \geq \bar{\rho}$ holds, then $\hat{x}^{acc}$ is an approximate solution to the problem (13) with

$$\Psi_n(\hat{x}^{acc}) \leq \Psi^* + \limsup_{\lambda \in I} \hat{\theta}_k(\lambda) + \limsup_{\lambda \in I} \left( \max_{l \in I_\eta} \right),$$

where $\Psi^*$ is the optimal value of function $\Psi_n$.\n
Suppose that the serious point sequence finally stabilizes, i.e., there exists a constant $m$ such that Algorithm 1 is well defined. In the next section, we will study separately the last point of the restart steps if the number of noise management steps is finite, there exists some iterative index $\hat{n}$ that $\{\hat{x}^{(\hat{n})}\}$ steps or null steps are done for $\{\{\hat{x}^{(\hat{n})}\} \}$.

Moreover, for any cluster point $\hat{x}^{\text{acc}}$ of $\{\hat{x}^{(\hat{n})}\}$, passing to the limit to (2a), we obtain

$$\Psi_n(\hat{x}^{\text{acc}}) - \limsup_{\lambda \to \infty} \theta_{k(\lambda)} \leq \lim_{\lambda \to \infty} (\hat{f}^{(\lambda)} + h(\hat{x}^{(\lambda)})).$$

Rewriting the two inequalities above, the conclusion holds. $\square$

Note that by the definition of function $\Psi_n$ and large enough index $n$, we have $\Psi_n(x) = \psi(x) + \frac{\eta}{2} \|x - \hat{x}^{\text{acc}}\|^2$ and $\Psi_n(\hat{x}^{\text{acc}}) = \psi(\hat{x}^{\text{acc}})$. By the above discussions, we have $\psi(\hat{x}^{\text{acc}}) \leq \psi^* + \frac{\eta}{2} \|x^* - \hat{x}^{\text{acc}}\|^2 + \limsup_{\lambda \in \mathcal{I}} \theta_{k(\lambda)} + \limsup_{\lambda \in \mathcal{I}} \max_{l \in I_{k(\lambda)}} \epsilon_l$ where $x^*$ and $\psi^*$ are local optimal solution and optimal value respectively. Then, $\hat{x}^{\text{acc}}$ is an approximate solution to the primal problem (1). There are some corollaries from Lemma 5, which are much important for convergent analysis. We state it here but omit its proof.

**Corollary 1.** (i) If for some iteration index $\lambda$, $\eta_{\lambda} \geq \hat{\rho}$ holds and the optimality measure satisfies $V_{\lambda} = 0$, then the serious point $\hat{x}^{(\lambda)}$ is an approximate solution to problem (13) with

$$\Psi_n(\hat{x}^{(\lambda)}) \leq \psi^* + \theta_{k(\lambda)} + \max_{l \in I_{k(\lambda)}} \epsilon_l. \tag{28}$$

(ii) Suppose that the serious point sequence finally stabilizes, i.e., there exists a constant $m$ such that for all $\lambda \geq m$, we have $\hat{x}^{(\lambda)} = \hat{x}^{(m)}$. If $\eta_m \geq \hat{\rho}$ holds, then $\hat{x}^{(m)}$ is an approximate solution to the problem (13) with

$$\Psi_n(\hat{x}^{(m)}) \leq \psi^* + \theta_{k(m)} + \max_{l \in I_{k(m)}} \epsilon_l. \tag{29}$$

Note that if an very large loop of noise management happens after some iteration $\hat{I}$ and $\eta_{\hat{I}} \geq \hat{\rho}$, then the proximal center keeps unchanged. According to (29), the last serious point $\hat{x}^{(\hat{I})}$ is an approximate solution to problem (13). From above lemmas and corollary, we have Algorithm 1 is well defined. In the next section, we will study separately the last two cases.

**4. Convergence Theory**

In this section, we study separately the last two cases above. The similar proof process can be found in [13,16,17,38,40]. In the following lemma, the second case, i.e., finite serious step with very large null steps, is considered.

**Lemma 6.** Assumption 1 holds and suppose that, after some iteration $\hat{n}$, $\eta_{\hat{n}} \geq \hat{\rho}$ holds and there is no serious step declared in Algorithm 1. Then, there exists a subsequence $\{x^n\}_{n \in \mathcal{I}_n}$, such that $V_n \to 0$ as $\mathcal{I}_n \ni n \to \infty$.

**Proof.** After some iteration $\hat{n}$, no serious step is declared. Hence either noise management steps or null steps are done for $n \geq \hat{n}$. The serious point does not change, i.e., for all $n \geq \hat{n}$, $\hat{x}^{(n)} = \hat{x}^{(\hat{n})}$. For notational simplicity, we denote $\hat{x} := \hat{x}^{(\hat{n})}$.

If the number of noise management steps is very large, we have that $\mu_n \to 0$ as $\mathcal{I}_n \ni n \to \infty$. The previous proof indicates that there exists a subsequence $\{x^{n+1}\}_{n \in \mathcal{I}_n}$ such that $V_n \to 0$ as $\mathcal{I}_n \ni n \to \infty$.

Suppose there is only a finite number of noise management steps. Since the number of the restart steps is finite, there exists some iterative index $\hat{n}$, such that (22) holds.
only null steps occur for all $n \geq \hat{n}$. Consequently, $\{\mu_n\}$ is a nondecreasing sequence since $\mu_{n+1} \in [\gamma \mu_n, \mu_{\text{max}}]$ for all $n > \hat{n}$. Meanwhile $\mu_n \to \bar{\mu} \leq \mu_{\text{max}}$ as $n \to \infty$. In the following, we show $\delta_n \to 0$. Let $P_n$ be the partial linearization of the the QP model (15), that is,

$$P_n(x) := \Phi_n^{\text{lin}}(x) + \frac{\mu_n}{2} \|x - \hat{x}\|^2, \quad \forall x \in \mathcal{T}.$$  

By Lemma 10.10 in [39], we know that the rules to apply selection on the bundles guarantee that $\Phi_n^{\text{lin}}(x) \leq \Phi_{n+1}(x)$ holds. By the inequality (8), we have

$$P_n(\hat{x}) = \Phi_n^{\text{lin}}(\hat{x}) \leq \psi(\hat{x}) + \frac{\eta_n}{2} \|\hat{x} - \hat{x}\|^2 + \hat{\varepsilon} = \psi(\hat{x}) + \hat{\varepsilon}. \quad (30)$$

Similarly, evaluating $P_n$ at $x^{n+2}$, and using the fact that $\mu_{n+1} \geq \gamma \mu_n$, we have

$$P_n(x^{n+2}) \leq \Phi_{n+1}(x^{n+2}) + \frac{\mu_{n+1}}{2} \|x^{n+2} - \hat{x}\|^2$$

$$= \Phi_{n+1}^{\text{lin}}(x^{n+2}) - (G^{n+1}, x^{n+2} - x^{n+2}) + \frac{\mu_{n+1}}{2} \|x^{n+2} - \hat{x}\|^2 = P_{n+1}(x^{n+2}).$$

Furthermore, $x^{n+1}$ is the unique solution to (15), then $\nabla P_n(x^{n+1}) = 0$. By Taylor’s expansion, we get

$$P_n(x) = P_n(x^{n+1}) + \frac{\mu_n}{2} \|x - x^{n+1}\|^2.$$

Hence the following two equalities hold

$$\begin{cases}
P_n(x^{n+2}) = P_n(x^{n+1}) + \frac{\mu_n}{2} \|x^{n+2} - x^{n+1}\|^2, \\
P_n(\hat{x}) = P_n(x^{n+1}) + \frac{\mu_n}{2} \|\hat{x} - x^{n+1}\|^2.
\end{cases}$$

Using the relationship above, the fact $\mu_n \geq \mu_\hat{n}$ and (30), we obtain

$$\begin{cases}
P_n(x^{n+1}) + \frac{\mu_n}{2} \|x^{n+2} - x^{n+1}\|^2 \leq P_{n+1}(x^{n+2}), \\
P_n(x^{n+1}) + \frac{\mu_n}{2} \|\hat{x} - x^{n+1}\|^2 = P_n(\hat{x}) \leq \psi(\hat{x}) + \hat{\varepsilon}.
\end{cases} \quad (31)$$

Then, the sequence $\{P_n(x^{n+1})\}_{n \geq \hat{n}}$ is nondecreasing and bounded. Hence the limit exists:

$$P_n(x^{n+1}) \to P^* < \infty, \text{ and } \|x^{n+1} - x^\ast\| \to 0, \text{ when } n \to \infty. \quad (32)$$

Then, the sequence of null steps $\{\delta_n\}$ is bounded. By (16) and $\{\mu_n\}$ is bounded, we have $\{G^{n+1}\}$ is bounded (see [39]). Since for $n > \hat{n}$, the serious steps test is not satisfied, so by the definition of $\delta_{n+1}$, we have

$$\hat{j}^{n+1} + h(x^{n+1}) + \eta_n \|x^{n+1} - \hat{x}\|^2 - \Phi_n(x^{n+1}) > (1 - m_1) \delta_{n+1}.$$  

Since $\hat{j}^{n+1} + h(x^{n+1}) + \frac{\eta_n}{2} \|x^{n+1} - \hat{x}\|^2 \leq \Phi_{n+1}(x^{n+2}) + (G^{n+1}, x^{n+1} - x^{n+2})$ holds, then by the definition of partial linearization and $\Omega := \hat{j}^{n+1} + h(x^{n+1}) + \eta_n \|x^{n+1} - \hat{x}\|^2 - \Phi_n(x^{n+1})$, we have

$$\Omega \leq \Phi_{n+1}(x^{n+2}) - \Phi_n(x^{n+1}) + \|G^{n+1}\| \|x^{n+1} - x^{n+2}\| + \frac{\eta_n}{2} \|x^{n+1} - \hat{x}\|^2$$

$$= \Phi_{n+1}^{\text{lin}}(x^{n+2}) - \Phi_n^{\text{lin}}(x^{n+1}) + \|G^{n+1}\| \|x^{n+1} - x^{n+2}\| + \frac{\eta_n}{2} \|x^{n+1} - \hat{x}\|^2$$

$$= P_{n+1}(x^{n+2}) - P_n(x^{n+1}) + \|G^{n+1}\| \|x^{n+1} - x^{n+2}\| - \frac{\mu_{n+1}}{2} \|x^{n+2} - \hat{x}\|^2 + \frac{\mu_n}{2} \|x^{n+1} - \hat{x}\|^2 - \frac{\eta_n}{2} \|x^{n+1} - \hat{x}\|^2.$$
By (32), Theorem 1 in [16] and $\mu_n \to \bar{\mu} \leq \mu_{\text{max}}$, we have the right side of the above inequality vanishes as $n \to \infty$. So $\Omega \to 0$ holds as $n \to \infty$. Hence

$$0 \leq (1 - m_1) \delta_{n+1} < \hat{\beta}^{n+1} + h(x^{n+1}) + \eta_n \|x^{n+1} - \bar{x}\|^2 - \Phi_n(x^{n+1}) \to 0.$$ 

Then, $\delta_{n+1} \to 0$ holds as $n \to \infty$. By (24), $V_n \to 0$ holds as $\bar{l} < n \to \infty$. \hfill \Box

**Theorem 1.** Suppose Algorithm 1 loops forever and Assumption 1 holds. Assume there are finite number of serious steps and $\eta \geq \bar{\rho}$. Then, the last serious point $\bar{x}$ is an approximate solution of problem (13) with

$$\Psi_n(\bar{x}) \leq \Psi_n^* + \theta_{\bar{x}} + \limsup_{n \to \infty} \max_{l \in I_n} \varepsilon_l.$$

**Proof.** (i) If very large noise management steps happen, Algorithm 1 finally stops and the conclusion holds. (ii) If very large null steps happen in Algorithm 1, by Lemma 6 and the second term of Corollary 1, the conclusion holds. \hfill \Box

The case of very large serious points generated in Algorithm 1 is considered in the next lemma. For notational convenience, we denote by $K$ the subset of iterations which are chosen as serious points. Let $\bar{x}^k$ and $\bar{x}^{k+}$ be two successive serious points.

**Lemma 7.** Suppose an very large sequence of serious steps is generated in Algorithm 1, Assumption 1 and $\eta \geq \bar{\rho}$ hold. Then, the $V_n \to 0$ as $\bar{K} \ni n \to \infty$.

**Proof.** Since the serious points satisfy the descent condition (25), for two successive serious points $\bar{x}^k$ and $\bar{x}^{k+}$, applying the descent condition inequality, we have

$$\bar{f}^k + h(\bar{x}^k) - \bar{f}^{k+} - h(\bar{x}^{k+}) - \frac{\eta_m}{2} \|\bar{x}^{k+} - \bar{x}^k\|^2 \geq m_1 \delta_{n_k}.$$ 

Rewriting the above inequality, we have

$$\bar{f}^k + h(\bar{x}^k) - \bar{f}^{k+} - h(\bar{x}^{k+}) \geq m_1 \delta_{n_k} + \frac{\eta_m}{2} \|\bar{x}^{k+} - \bar{x}^k\|^2 > 0.$$ 

Then, we have the sequence $\{\bar{f}^k + h(\bar{x}^k)\}$ is strictly decreasing. By summing up this inequality for all serious steps, we deduce that

$$\frac{1}{2} \sum_{k \in K} \eta_k \|\bar{x}^{k+} - \bar{x}^k\|^2 + m_1 \sum_{k \in K} \delta_k < \infty, \text{ with } \delta_k > 0.$$ 

Hence, the above inequality deduces $\delta_k \to 0$. Since (22) holds, then by (24), we have $V_k \to 0$ as $\bar{K} \ni k \to \infty$. \hfill \Box

**Theorem 2.** Suppose Algorithm 1 loops forever, there are very large number of serious steps and $\eta \geq \bar{\rho}$ holds. Then, any accumulation point $\bar{x}_{\text{acc}}$ of serious points sequence $\{\bar{x}^k\}_{k \in K}$ is an approximate solution of the problem (13) with

$$\Psi_n(\bar{x}_{\text{acc}}) \leq \Psi_n^* + \limsup_{K \ni k \to \infty} \theta_{\bar{x}_k} + \limsup_{K \ni k \to \infty} \max_{l \in I_n} \varepsilon_l.$$ 

**Proof.** The conclusion follows from Lemma 5 and Lemma 7. \hfill \Box

5. Numerical Results

In this section, we consider two Ferrier polynomial functions (see [10,15,16]) and some DC (difference of convex) functions (see [41–44]). The section is divided into three parts. We code Algorithm 1 in MATLAB R2016 and run it on a PC with 2.10 GHZ CPU. Meanwhile, the Quadratic programming solver for Algorithm 1 in this paper is the Quadprog.m, which
is available in the Optimization Toolbox in MATLAB. Note that the quadratic programming solver is not special and any solver for quadratic programming is accepted.

5.1. Two Polynomial Functions

In this subsection, we first present two polynomial functions which are in the form of the objective function (1). The two polynomial functions have the following forms:

$$\psi_1(x) := \sum_{i=1}^{N} |\omega_i(x)| + \frac{\|x\|^2}{2}, \quad \psi_2(x) := \sum_{i=1}^{N} |\omega_i(x)| + \frac{\|x\|^2}{2},$$

where $\omega_i(x) : \mathbb{R}^N \rightarrow \mathbb{R}$ is defined by $\omega_i(x) = (ix_i^2 - 2x_i) + \sum_{j=1}^{N} x_j$, where $\forall x \in \mathbb{R}^N$ and for each $i = 1, \ldots, N$. It is clear that the above functions are nonconvex, nonsmooth, lower-$C^2$ and have 0 as their global minimizer. If we denote $h(x) = \sum_{i=1}^{N} |\omega_i(x)|$ and $f(x) = \|x\|^2/2$ or $f(x) = \|x\|/2$, the above functions are clear in the form (1). In the following, we adopt initial points $x_0 = [1, 1, \ldots, 1]$ and consider the case $N \in \{1, 2, \ldots, 19, 20\}$. The parameters in this subsection are set as follows: $m_0 = 0.01$, $\kappa = 0.9$, $R_0 = 10$, $M_0 = 10$, $\tau = 2$, $\gamma = 2$, $\mu_{\max} = 10^2$ and $Tol = 10^{-6}$. We also stop the algorithm when the iterative number is over 1000. First, we present the numerical results in Tables 1 and 2 for the case $\theta = 0$ and $\varepsilon = 0$, that is the exact case, and compare them with the results in [16]. We call the algorithm in [16] as the RedistProx algorithm. Meanwhile, we adopt $I_{n} = \{0, 1, 2, \ldots\}$. Note that in the exact case, we stop the progress when $\delta_k \leq Tol$ occurs, which is the same in [16]. In exact case, the linearization error $e^k_{f,i}$ and $e^k_{\psi,i}$ are nonnegative, so the noise attenuation steps never happen. Then, in the numerical results for exact cases, the NNA is always zero, and we omit the NNA in the Tables 1 and 2. The columns of Tables have the following meanings: Dim: the the tested problem dimension, NS: the number of serious steps, NNA: the number of noise attenuation steps, NF: the number of oracle function evaluations used, $f_k$: the minimal function value found, $\delta_k$: the value of $\delta_k$ at the final iteration, $\psi^*$: the optimal function found, $V_k$: the value of $V_k$ at the final iteration, RN: the number of restart steps, Nu: the number of null steps.

From Tables 1 and 2, in most cases, our algorithm has a higher accuracy and compares with the RedistProx algorithm in [16]. For $N = \{11, 12, 13, 14\}$, we adopt a larger initial model prox-parameter $\mu_0$ (a smaller steplength), $R_0 = 1000$. Meanwhile, the Tol and other parameters keep unchanged. For $N \in \{15, 16, 17, 18, 19, 20\}$, We take $Tol = 10^{-5}$, $R_0 = 1000$ and keep other parameters unchanged. The numerical results for $\psi_1(x)$ and $\psi_2(x)$ are reported in Table 3.

From Table 3, Algorithm 1 can solve the two Ferrier polynomial functions in higher dimension successfully and have a reasonable and higher accuracy. The parameters $\mu$ and $\eta$ eventually keep unchanged in exact case, which is illustrated by Figure 1.

**Table 1.** The numerical results of Algorithm 1 and RedistProx for $\psi_1(x)$.

| Dim | Algorithm 1 NS | Algorithm 1 NF | Algorithm 1 $\delta_k$ | $f_k$ | RedistProx NF | RedistProx $\delta_k$ | $f_k$ | $\psi^*$ |
|-----|----------------|----------------|-------------------------|-------|----------------|-----------------------|-------|---------|
| 1   | 10             | 14             | 5                       | 1.0000 $\times$ 10^{-4} | 0.5000 | 0               |
| 2   | 20             | 31             | 3.0702 $\times$ 10^{-9} | 10    | 8.0199 $\times$ 10^{-4} | 3.6623 $\times$ 10^{-4} | 0     |
| 3   | 12             | 18             | 5.1480 $\times$ 10^{-8} | 12    | 2.5212 $\times$ 10^{-1} | 5.2922 $\times$ 10^{-2} | 0     |
| 4   | 11             | 19             | 7.0451 $\times$ 10^{-6} | 18    | 1.23599 $\times$ 10^{-1} | 2.5722 $\times$ 10^{-2} | 0     |
| 5   | 15             | 18             | 5.4569 $\times$ 10^{-2} | 26    | 3.04311 $\times$ 10^{-1} | 5.1166 $\times$ 10^{-2} | 0     |
| 6   | 70             | 375            | 1.7556 $\times$ 10^{-3} | 60    | 2.0000 $\times$ 10^{-5} | 0.0000000 | 0     |
| 7   | 29             | 73             | 1.2567 $\times$ 10^{-2} | 34    | 2.57493 $\times$ 10^{-1} | 2.39281 $\times$ 10^{-1} | 0     |
| 8   | 40             | 168            | 1.0969 $\times$ 10^{-7} | 56    | 1.54684 $\times$ 10^{-1} | 6.9825 $\times$ 10^{-2} | 0     |
| 9   | 31             | 54             | 3.7028 $\times$ 10^{-7} | 150   | 1.4000 $\times$ 10^{-5} | 0.0000000 | 0     |
| 10  | 41             | 70             | 1.4399 $\times$ 10^{-7} | 61    | 1.71962 $\times$ 10^{-1} | 2.17352 $\times$ 10^{-1} | 0     |
Next, inexact data are referred and we consider the cases of random noises for function value and subgradient. We introduce two kinds of random noises in matlab codes. The first case is \( \theta_1 = 0.01 \times \text{normrnd}(0,0.1) \), and \( \epsilon_j = 0.01 \times \text{normrnd}(0,0.1,1,dim) \). The code \( \text{normrnd}(0,0.1,1,dim) \) generates random numbers from the normal distribution with mean value 0, standard deviation 0.1 and scalars \( \text{dim} \) and 1 are the row and column dimensions. We take \( n_1 = 0.01, \kappa = 0.9, \gamma = 2, \tau = 2, M_0 = 5 \) and \( R_0 = 1000 \) in this random error case. The algorithm stops when \( V_k \leq \text{Tol} \) holds or the number of function evaluated is over 1000. The numerical results for this case are report in Table 4.

Table 2. The numerical results of Algorithm 1 and RedistProx for \( \psi_2(x) \).

| Dim | Algorithm 1 | RedistProx |
|-----|-------------|------------|
|     | NS | NF | \( \delta_k \) | fk | NS | NF | \( \delta_k \) | fk | \( \psi^* \) |
| 1   |   |   | 9.8111 \times 10^{-9} | 0.5000 | 13 | 0.0000 | 7.0000 \times 10^{-6} | 0 |
| 2   |   |   | 7.3850 \times 10^{-7} | 3.6061 \times 10^{-9} | 16 | 0.0000 | 0.0000 | 0 |
| 3   |   |   | 9.9438 \times 10^{-7} | 5.8183 \times 10^{-7} | 17 | 2.0000 \times 10^{-6} | 0 |
| 4   |   |   | 8.7577 \times 10^{-7} | 2.0964 \times 10^{-2} | 23 | 1.2899 \times 10^{-1} | 1.9105 \times 10^{-2} | 0 |
| 5   |   |   | 3.7942 \times 10^{-7} | 2.5534 \times 10^{-7} | 31 | 3.7467 \times 10^{-1} | 3.5133 \times 10^{-1} | 0 |
| 6   |   |   | 8.7508 \times 10^{-7} | 1.4844 \times 10^{-7} | 35 | 2.1773 \times 10^{-1} | 1.1383 \times 10^{-1} | 0 |
| 7   |   |   | 4.6810 \times 10^{-7} | 4.5559 \times 10^{-2} | 41 | 1.8208 \times 10^{-1} | 1.2009 \times 10^{-1} | 0 |
| 8   |   |   | 6.7910 \times 10^{-7} | 2.2678 \times 10^{-7} | 41 | 8.6662 \times 10^{-2} | 1.084600 | 0 |
| 9   |   |   | 9.0456 \times 10^{-7} | 2.7211 \times 10^{-7} | 42 | 2.3830 \times 10^{-1} | 7.8253 \times 10^{-1} | 0 |
| 10  |   |   | 9.6527 \times 10^{-7} | 2.8546 \times 10^{-7} | 66 | 8.7404 \times 10^{-2} | 3.6327 \times 10^{-2} | 0 |

Table 3. The numerical results of Algorithm 1 for \( \psi_1(x) \) and \( \psi_2(x) \).

| Dim | Algorithm 1 for \( \psi_1(x) \) | Algorithm 1 for \( \psi_2(x) \) |
|-----|---------------------------------|---------------------------------|
|     | \( \delta_k \) | fk | \( \delta_k \) | fk | \( \psi^* \) |
| 11  | 4.7172 \times 10^{-7} | 1.5830 \times 10^{-7} | 173 | 8.4825 \times 10^{-7} | 1.2989 \times 10^{-7} | 0 |
| 12  | 7.0756 \times 10^{-7} | 3.5108 \times 10^{-7} | 174 | 9.9598 \times 10^{-7} | 4.0815 \times 10^{-7} | 0 |
| 13  | 8.1686 \times 10^{-7} | 3.0480 \times 10^{-7} | 178 | 9.7867 \times 10^{-7} | 6.7149 \times 10^{-7} | 0 |
| 14  | 8.8377 \times 10^{-7} | 4.1342 \times 10^{-7} | 165 | 9.1131 \times 10^{-7} | 1.8555 \times 10^{-7} | 0 |
| 15  | 9.5086 \times 10^{-6} | 1.2213 \times 10^{-6} | 167 | 7.7433 \times 10^{-6} | 3.8891 \times 10^{-6} | 0 |
| 16  | 5.0300 \times 10^{-6} | 2.0755 \times 10^{-6} | 167 | 6.5387 \times 10^{-6} | 3.1144 \times 10^{-6} | 0 |
| 17  | 8.5472 \times 10^{-6} | 2.8304 \times 10^{-6} | 170 | 7.4924 \times 10^{-6} | 2.9795 \times 10^{-6} | 0 |
| 18  | 9.8221 \times 10^{-6} | 2.6160 \times 10^{-6} | 150 | 5.0522 \times 10^{-6} | 3.6197 \times 10^{-6} | 0 |
| 19  | 6.7693 \times 10^{-6} | 2.6364 \times 10^{-6} | 173 | 9.0980 \times 10^{-6} | 2.0003 \times 10^{-6} | 0 |
| 20  | 7.7570 \times 10^{-6} | 4.46795 \times 10^{-2} | 236 | 9.8828 \times 10^{-6} | 3.0570 \times 10^{-6} | 0 |

![Figure 1. Values of \( \eta \) and \( \mu \) along NI with \( n=17 \) for \( \psi_1(x) \)](image-url)
From Table 4, we have Algorithm 1 can solve $\psi_1(x)$ and $\psi_2(x)$ successfully for random errors in a reasonable accuracy. We also focus on the parameters $\eta$ and $\mu$ in the implementation of Algorithm 1. Although the convexification parameter $\eta_n$ eventually keeps unchanged, the update strategy of the proximal parameter $\mu_n$ is complicated. When the noise management step occurs, parameter $\mu_n$ is decreasing to reduce the ‘noise’ errors impact. When the unacceptable condition happens, we increase the parameter $\mu_n$ to get a smaller step length. Figure 2 shows the variation of parameters $\eta$ and $\mu$ along NF for $\psi_1(x)$ with $N = 19$ in normal random error case.

![Values of parameters $\eta$ and $\mu$ with n=19 for $\psi_1(x)$ in random case](image)

**Figure 2.** The values of $\eta$ and $\mu$ in function $\psi_1(x)$ with $N = 19$ in inexact case.

In the following, we introduce the error case of $\theta_j = 0.01 \ast \text{unifrnd}(0, 1)$ and $\varepsilon_j = 0.01 \ast \text{unifrnd}(0, 1, \text{dim})$. The code $\text{unifrnd}(0, 1, \text{dim})$ gives a similar case with the ‘normrd’ case. In this case, we adopt two $\text{Tol}$ values and two initial proximal parameter values $R_0$
for different dimension of the variables. Concretely, we take $m_1 = 0.01, \kappa = 0.9, \tau = 2, \gamma = 2, M_0 = 5$ and $R_0 = 20, Tol = 10^{-6}$ for $N \in \{1, 2, \cdots, 8\}$. For $N \in \{9, 10, \cdots, 19, 20\}$, we take $R_0 = 200, Tol = 10^{-3}$ and the other parameters keep unchanged. We also take 1000 as the upper limit of function evaluated. The algorithm stops when $V_k \leq Tol$ holds or the number of function evaluated is over 1000. The numerical results for this error case are reported in Table 5.

Table 5. The numerical results of Algorithm 1 for $\psi_1(x)$ and $\psi_2(x)$ in unifrnd error case.

| Dim | Nu | NS | NNA | NF | $V_k$ | $f_k$ | Dim | Nu | NS | NNA | NF | $V_k$ | $f_k$ |
|-----|----|----|-----|----|-------|-------|-----|----|----|-----|----|-------|-------|
| 1   | 4  | 9  | 83  | 97 | 9.0613e-7 | -6.4794e-3 | 2   | 0  | 19 | 22  | 9.7755e-7 | 4.9060e-1 |
| 2   | 3  | 17 | 127 | 148| 9.1482e-7 | -1.8758e-4 | 1   | 18 | 79 | 99  | 9.0223e-7 | 7.7941e-4 |
| 3   | 3  | 11 | 110 | 125| 9.0388e-7 | 1.7436e-7  | 2   | 11 | 83 | 97  | 9.5395e-7 | -7.1758e-4 |
| 4   | 2  | 11 | 79  | 93 | 9.4813e-7 | 7.1079e-4  | 0   | 10 | 81 | 92  | 9.4233e-7 | 2.0590e-3 |
| 5   | 1  | 10 | 96  | 108| 9.4407e-7 | -5.4571e-4 | 3   | 8  | 102| 114 | 9.0188e-7 | 6.8412e-3 |
| 6   | 4  | 11 | 102 | 118| 9.9976e-7 | 7.3570e-3  | 3   | 13 | 72 | 89  | 9.4247e-7 | 6.0276e-3 |
| 7   | 6  | 10 | 112 | 129| 9.2399e-7 | 7.5234e-2  | 7   | 9  | 85 | 102 | 9.2723e-7 | 1.2366e-3 |
| 8   | 7  | 16 | 108 | 132| 9.7060e-7 | 1.4824e-2  | 6   | 13 | 102| 122 | 9.1341e-7 | 7.7913e-3 |
| 9   | 5  | 38 | 81  | 125| 9.3352e-7 | 5.2611e-4  | 2   | 36 | 108| 147 | 9.0509e-5 | 1.4101e-2 |
| 10  | 7  | 36 | 94  | 138| 9.3808e-5 | 2.7920e-3  | 2   | 36 | 107| 146 | 9.9270e-5 | 2.3615e-2 |
| 11  | 4  | 36 | 125 | 166| 9.3952e-5 | 6.2396e-4  | 2   | 34 | 142| 179 | 9.7926e-5 | 2.2681e-2 |
| 12  | 32 | 39 | 249 | 321| 9.8977e-5 | 1.1030e-3  | 2   | 34 | 96 | 133 | 9.9074e-5 | 1.3646e-2 |
| 13  | 7  | 36 | 93  | 137| 9.0263e-5 | 3.8576e-2  | 6   | 34 | 103| 144 | 9.7738e-5 | 4.7721e-2 |
| 14  | 31 | 41 | 231 | 304| 9.6781e-5 | 4.2488e-3  | 0   | 40 | 83 | 124 | 9.6485e-5 | 7.3474e-3 |
| 15  | 2  | 38 | 110 | 151| 9.5999e-5 | 6.1932e-2  | 7   | 34 | 104| 146 | 9.4630e-5 | 4.2409e-2 |
| 16  | 5  | 37 | 92  | 135| 8.6662e-5 | 4.9052e-2  | 6   | 34 | 106| 147 | 9.4022e-5 | 1.1794e-1 |
| 17  | 24 | 46 | 205 | 276| 9.5651e-5 | -8.0278e-4 | 4   | 34 | 116| 155 | 9.9456e-5 | 5.7203e-2 |
| 18  | 32 | 42 | 217 | 292| 9.4322e-5 | 1.4177e-3  | 4   | 33 | 107| 145 | 9.8050e-5 | 5.2519e-2 |
| 19  | 12 | 38 | 123 | 174| 9.7789e-5 | 1.1508e-1  | 5   | 35 | 126| 167 | 9.2005e-5 | 2.8864e-1 |
| 20  | 24 | 52 | 118 | 195| 9.8336e-5 | 7.6241e-3  | 21  | 44 | 85 | 151 | 9.3581e-5 | 9.4419e-2 |

From Table 5, Algorithm 1 can solve $\psi_1(x)$ and $\psi_2(x)$ in a reasonable accuracy for the ‘unifrnd’ random error case. In this inexact case, we also illustrate the vary of $\eta_n$ and $\mu_n$ in Figures 3 and 4. The parameter $\eta_n$ is eventually stable. Although the vary of the proximal parameter $\mu_n$ is complicate for inexact case, the hypothesis about the upper limits for $\mu_n$ in the numerical experiment is reasonable, which are illustrated in the numerical testing.

![Values of parameters $\eta$ and $\mu$ for $\psi_1(x)$ with $n=14$ for 'unifrnd' error case](image)

Figure 3. Values of $\eta$ and $\mu$ in function $\psi_1(x)$ with $N = 14$ in unifrnd error case.
5.2. Noise’s Impact on Solution Accuracy

The error have different types. To analysis the impact of different noise types, we test five different types of inexact oracles:

- **NNE** (no noise error): in this case, $\bar{\theta} = \bar{\varepsilon} = 0$ and $\theta_i = \varepsilon_i = 0$ for all $i$ in iterative process;
- **CNE** (constant noise error): in this case, $\bar{\theta} = \bar{\varepsilon} = \theta_i = \varepsilon_i = 0.01$, for all $i$ in iterative process;
- **VNE** (vanishing noise error): in this case, $\bar{\theta} = \bar{\varepsilon} = 0.01$ and $\theta_i = \varepsilon_i = \min\{0.01, \|x^i\|/100\}$ for all $i$ in iterative process;
- **CGNE** (constant subgradient noise error): in this case, $\bar{\theta} = \theta_i = 0$ and $\bar{\varepsilon} = \varepsilon_i = 0.01$ for all $i$ in iterative process;
- **VGNE** (vanishing subgradient noise error): in this case, we set $\bar{\theta} = \theta_i = 0$ and $\varepsilon_i = \min\{0.01, \|x^i\|/100\}$ for all $i$ in iterative process;

In the numerical experiment, the parameters involved are the same with that in the ‘unifrnd’ error case. We present the numerical results of no noise error case (exact values case) in Table 6. In the test, for $N = 2$, we take $Tol = 10^{-5}$. In exact case, the number of NNA is always 0, then we omit the columns of NNA in Table 6.

In the following, we present the numerical results of constant noise error case in Table 7. The parameters are same with that in the NNE case except that for $\psi_1(x)$ with $N = 2, 18$. For the case of $\psi_1(x)$ with $N = 2$, we take $Tol = 10^{-1}$, $R_0 = 200$ and the other parameters keep unchanged. For the case of $\psi_1(x)$ with $N = 18$, we take $Tol = 10^0$, $R_0 = 200$ and the other parameters keep unchanged. For the case of $\psi_2(x)$ with $N = 8$, we take $Tol = 10^0$, $R_0 = 500$. Table 8 presents the results for the vanishing noise error case. The parameters keep unchanged except that for $\psi_1(x)$ with $N = 2, 7, 14, 18$. In these cases, we take $Tol = 10^{-1}$, $R_0 = 200$ and other parameters keep unchanged.

In the following, Table 9 presents the results for the constant subgradient noise error case (CGNE). The parameters keep unchanged except the case of $\psi_1(x)$ with $N = 18$. In this case, we take $Tol = 10^0$ and $R_0 = 500$. Table 10 presents the results for the vanishing subgradient noise error case (VGNE). The parameters keep unchanged except the cases of $\psi_1(x)$ with $N = 2, 7, 14, 18$. In these cases, we take $Tol = 10^{-1}$, $R_0 = 200$ and other parameters keep unchanged.
### Table 6. The numerical results of Algorithm 1 for $\psi_1(x)$ and $\psi_2(x)$ in NNE case.

| Dim | Algorithm 1 for $\psi_1(x)$ | Algorithm 1 for $\psi_2(x)$ |
|-----|-------------------------------|-------------------------------|
|     | $V_k$                        | $V_k$                        | $f_k$                     | $f_k$                     |
| 1   | $5.425 \times 10^{-7}$        | $9.296 \times 10^{-7}$       | 2                          | 2                          |
| 2   | $10.76 \times 10^{-8}$        | $16.50 \times 10^{-8}$       | 7                          | 7                          |
| 3   | $19.27 \times 10^{-9}$        | $30.68 \times 10^{-9}$       | 12                         | 12                         |
| 4   | $39.62 \times 10^{-10}$       | $60.96 \times 10^{-10}$      | 16                         | 16                         |
| 5   | $79.48 \times 10^{-11}$       | $120.73 \times 10^{-11}$     | 20                         | 20                         |
| 6   | $126.25 \times 10^{-12}$      | $198.43 \times 10^{-12}$     | 24                         | 24                         |
| 7   | $183.51 \times 10^{-13}$      | $280.67 \times 10^{-13}$     | 28                         | 28                         |
| 8   | $240.77 \times 10^{-14}$      | $360.99 \times 10^{-14}$     | 32                         | 32                         |
| 9   | $298.04 \times 10^{-15}$      | $448.27 \times 10^{-15}$     | 36                         | 36                         |
| 10  | $355.32 \times 10^{-16}$      | $535.54 \times 10^{-16}$     | 40                         | 40                         |

### Table 7. The numerical results of Algorithm 1 for $\psi_1(x)$ and $\psi_2(x)$ in CNE case.

| Dim | Algorithm 1 for $\psi_1(x)$ | Algorithm 1 for $\psi_2(x)$ |
|-----|-------------------------------|-------------------------------|
|     | $V_k$                        | $V_k$                        | $f_k$                     | $f_k$                     |
| 1   | $5.4925 \times 10^{-7}$       | $9.296 \times 10^{-7}$       | 1                          | 1                          |
| 2   | $10.76 \times 10^{-8}$        | $16.50 \times 10^{-8}$       | 2                          | 2                          |
| 3   | $19.27 \times 10^{-9}$        | $30.68 \times 10^{-9}$       | 3                          | 3                          |
| 4   | $39.62 \times 10^{-10}$       | $60.96 \times 10^{-10}$      | 4                          | 4                          |
| 5   | $79.48 \times 10^{-11}$       | $120.73 \times 10^{-11}$     | 5                          | 5                          |
| 6   | $126.25 \times 10^{-12}$      | $198.43 \times 10^{-12}$     | 6                          | 6                          |
| 7   | $183.51 \times 10^{-13}$      | $280.67 \times 10^{-13}$     | 7                          | 7                          |
| 8   | $240.77 \times 10^{-14}$      | $360.99 \times 10^{-14}$     | 8                          | 8                          |
| 9   | $298.04 \times 10^{-15}$      | $448.27 \times 10^{-15}$     | 9                          | 9                          |
| 10  | $355.32 \times 10^{-16}$      | $535.54 \times 10^{-16}$     | 10                         | 10                         |
Table 8. The numerical results of Algorithm 1 for $\psi_1(x)$ and $\psi_2(x)$ in VNE case.

| Dim | Nu | NS | NNA | NF | $V_k$ | fk |
|-----|----|----|-----|----|-------|----|
| 1   | 3  | 63 | 0   | 67 | $2.2411 \times 10^{-7}$ | $2.7773 \times 10^{-9}$ |
| 2   | 1  | 27 | 0   | 29 | $9.8574 \times 10^{-2}$ | $9.6203 \times 10^{-4}$ |
| 3   | 0  | 23 | $1.3800 \times 10^{-7}$ | 5 | $15.00 \times 10^{-5}$ | $4.5675 \times 10^{-7}$ |
| 4   | 6  | 20 | $3.5958 \times 10^{-7}$ | 16 | $20.00 \times 10^{-7}$ | $3.7227 \times 10^{-7}$ |
| 5   | 45  | 32 | $1.0979 \times 10^{-7}$ | 16 | $20.00 \times 10^{-7}$ | $9.0282 \times 10^{-7}$ |
| 6   | 11  | 22 | $1.0183 \times 10^{-7}$ | 34 | $25.00 \times 10^{-7}$ | $6.8176 \times 10^{-7}$ |
| 7   | 11  | 45 | $8.0881 \times 10^{-7}$ | 71 | $33.00 \times 10^{-7}$ | $9.4347 \times 10^{-7}$ |
| 8   | 88  | 44 | $2.4473 \times 10^{-7}$ | $3.16 \times 10^{-1}$ | $121.00 \times 10^{-7}$ | $7.9224 \times 10^{-7}$ |
| 9   | 32  | 79 | $2.4391 \times 10^{-6}$ | 20 | $59.00 \times 10^{-7}$ | $7.8092 \times 10^{-6}$ |
| 10  | 72  | 85 | $2.2020 \times 10^{-5}$ | 14 | $54.00 \times 10^{-6}$ | $8.7709 \times 10^{-5}$ |
| 11  | 91  | 80 | $1.7416 \times 10^{-6}$ | 21 | $64.00 \times 10^{-6}$ | $5.3167 \times 10^{-6}$ |
| 12  | 252 | 97 | $3.9407 \times 10^{-6}$ | 96 | $78.00 \times 10^{-6}$ | $8.7760 \times 10^{-6}$ |
| 13  | 39  | 75 | $4.3186 \times 10^{-6}$ | 17 | $51.00 \times 10^{-6}$ | $7.3807 \times 10^{-6}$ |
| 14  | 391 | 98 | $5.7300 \times 10^{-6}$ | 22 | $59.00 \times 10^{-6}$ | $9.4783 \times 10^{-6}$ |
| 15  | 75  | 105 | $2.5624 \times 10^{-5}$ | 290 | $85.00 \times 10^{-6}$ | $3.9385 \times 10^{-5}$ |
| 16  | 25  | 55 | $2.9491 \times 10^{-6}$ | 20 | $61.00 \times 10^{-6}$ | $7.7500 \times 10^{-6}$ |
| 17  | 12  | 63 | $4.1578 \times 10^{-5}$ | 35 | $57.00 \times 10^{-6}$ | $9.9252 \times 10^{-5}$ |
| 18  | 203 | 138 | $4.5494 \times 10^{-6}$ | 25 | $52.00 \times 10^{-6}$ | $6.6201 \times 10^{-6}$ |
| 19  | 88  | 96  | $4.5004 \times 10^{-6}$ | 46 | $401.00 \times 10^{-6}$ | $3.6082 \times 10^{-5}$ |
| 20  | 140 | 91  | $5.5017 \times 10^{-3}$ | 23 | $94.00 \times 10^{-3}$ | $9.3351 \times 10^{-5}$ |

Table 9. The numerical results of Algorithm 1 for $\psi_1(x)$ and $\psi_2(x)$ in CGNE case.

| Dim | Nu | NS | NNA | NF | $V_k$ | fk |
|-----|----|----|-----|----|-------|----|
| 1   | 2  | 61 | 10  | 74 | $5.4925 \times 10^{-7}$ | $1.3472 \times 10^{-7}$ |
| 2   | 215 | 31 | 90  | 337 | $4.3225 \times 10^{-5}$ | $7.1521 \times 10^{-7}$ |
| 3   | 2  | 13 | 62  | 78 | $9.9421 \times 10^{-7}$ | $5.2650 \times 10^{-5}$ |
| 4   | 2  | 12 | 42  | 57 | $9.0680 \times 10^{-7}$ | $6.7504 \times 10^{-4}$ |
| 5   | 5  | 15 | 10  | 31 | $9.2336 \times 10^{-7}$ | $4.6504 \times 10^{-4}$ |
| 6   | 16  | 23 | 27  | 67 | $8.4599 \times 10^{-7}$ | $3.1286 \times 10^{-6}$ |
| 7   | 162 | 53 | 23  | 239 | $7.8402 \times 10^{-7}$ | $2.3823 \times 10^{-6}$ |
| 8   | 11  | 27 | 35  | 74 | $9.6865 \times 10^{-7}$ | $9.8010 \times 10^{-6}$ |
| 9   | 10  | 56 | 6   | 73 | $7.9410 \times 10^{-5}$ | $4.2731 \times 10^{-6}$ |
| 10  | 76  | 84 | 0   | 161 | $6.2920 \times 10^{-5}$ | $1.9282 \times 10^{-6}$ |
| 11  | 47  | 69 | 0   | 117 | $8.3546 \times 10^{-5}$ | $3.7396 \times 10^{-6}$ |
| 12  | 363 | 104 | 468 | $9.4169 \times 10^{-5}$ | $2.6927 \times 10^{-6}$ |
| 13  | 52  | 93 | 0   | 146 | $6.3104 \times 10^{-5}$ | $2.1522 \times 10^{-6}$ |
| 14  | 173 | 82 | 0   | 256 | $7.8317 \times 10^{-5}$ | $2.7201 \times 10^{-6}$ |
| 15  | 28  | 71 | 34  | 134 | $9.1862 \times 10^{-5}$ | $1.1654 \times 10^{-4}$ |
| 16  | 34  | 70 | 0   | 105 | $7.4213 \times 10^{-5}$ | $2.6541 \times 10^{-6}$ |
| 17  | 104 | 95 | 0   | 200 | $6.0123 \times 10^{-5}$ | $3.0115 \times 10^{-3}$ |
| 18  | 2  | 77 | 0   | 80 | $8.0272 \times 10^{-5}$ | $3.1333 \times 10^{-2}$ |
| 19  | 74  | 109 | 0  | 184 | $9.9172 \times 10^{-5}$ | $2.6683 \times 10^{-6}$ |
| 20  | 44  | 70 | 0   | 115 | $5.8482 \times 10^{-5}$ | $6.6515 \times 10^{-3}$ |
Table 10. The numerical results of Algorithm 1 for $\psi_1(x)$ and $\psi_2(x)$ in VGNE case.

| Dim | Nu | NS | NNA | NF | $V_k$ | $f_k$ | Algorithm 1 for $\psi_1(x)$ | Algorithm 1 for $\psi_2(x)$ |
|-----|----|----|-----|----|-------|-------|--------------------------|--------------------------|
| 1   | 3  | 63 | 0   | 67 | $4.3452 \times 10^{-7}$ | $1.0116 \times 10^{-8}$ | 2 | 0 | 20 | 23 | $9.2964 \times 10^{-7}$ | 0.5000 |
| 2   | 27 | 298| 0   | 326| $3.3338 \times 10^{-2}$ | $1.1757 \times 10^{-3}$ | 18 | 23 | 41 | 83 | $7.9777 \times 10^{-7}$ | $2.7914 \times 10^{-6}$ |
| 3   | 6  | 17 | 0   | 24 | $4.6189 \times 10^{-7}$ | $4.8693 \times 10^{-3}$ | 2 | 17 | 2  | 22 | $7.6315 \times 10^{-7}$ | $1.3123 \times 10^{-7}$ |
| 4   | 2  | 20 | 0   | 23 | $3.4081 \times 10^{-7}$ | $6.7561 \times 10^{-8}$ | 3 | 15 | 38 | 57 | $9.4155 \times 10^{-7}$ | $9.1216 \times 10^{-6}$ |
| 5   | 32 | 32 | 0   | 65 | $9.6611 \times 10^{-7}$ | $1.3557 \times 10^{-7}$ | 5 | 16 | 45 | 67 | $8.0761 \times 10^{-7}$ | $2.2608 \times 10^{-5}$ |
| 6   | 16 | 24 | 0   | 41 | $3.8622 \times 10^{-7}$ | $9.8845 \times 10^{-8}$ | 5 | 21 | 58 | 85 | $9.2491 \times 10^{-7}$ | $2.4062 \times 10^{-5}$ |
| 7   | 6  | 57 | 0   | 64 | $2.9199 \times 10^{-2}$ | $1.0333 \times 10^{-5}$ | 8 | 20 | 68 | 97 | $9.4552 \times 10^{-7}$ | $4.4226 \times 10^{-3}$ |
| 8   | 130| 41 | 0   | 172| $5.8681 \times 10^{-7}$ | $4.3867 \times 10^{-8}$ | 7 | 15 | 93 | 116| $9.8671 \times 10^{-7}$ | $1.0319 \times 10^{-2}$ |
| 9   | 54 | 74 | 0   | 129| $8.5412 \times 10^{-5}$ | $2.3582 \times 10^{-6}$ | 8 | 54 | 19 | 82 | $8.9252 \times 10^{-5}$ | $1.3088 \times 10^{-5}$ |
| 10  | 91 | 79 | 0   | 171| $6.4727 \times 10^{-5}$ | $3.4788 \times 10^{-6}$ | 7 | 51 | 15 | 74 | $7.2476 \times 10^{-5}$ | $1.2106 \times 10^{-5}$ |
| 11  | 79 | 75 | 0   | 155| $7.3439 \times 10^{-5}$ | $4.7386 \times 10^{-6}$ | 6 | 54 | 36 | 97 | $4.8921 \times 10^{-5}$ | $7.0049 \times 10^{-5}$ |
| 12  | 172| 90 | 0   | 263| $4.2617 \times 10^{-5}$ | $3.9532 \times 10^{-6}$ | 9 | 70 | 39 | 119| $8.9773 \times 10^{-5}$ | $1.3272 \times 10^{-4}$ |
| 13  | 46 | 78 | 0   | 125| $8.4436 \times 10^{-5}$ | $2.9852 \times 10^{-6}$ | 5 | 50 | 40 | 96 | $9.4999 \times 10^{-5}$ | $2.1125 \times 10^{-4}$ |
| 14  | 8  | 78 | 0   | 87 | $6.3657 \times 10^{-2}$ | $7.5245 \times 10^{-6}$ | 9 | 47 | 34 | 96 | $9.2984 \times 10^{-5}$ | $7.1209 \times 10^{-5}$ |
| 15  | 203| 80 | 0   | 284| $6.9332 \times 10^{-5}$ | $1.4201 \times 10^{-6}$ | 49 | 80 | 54 | 184| $7.2507 \times 10^{-5}$ | $3.7361 \times 10^{-4}$ |
| 16  | 55 | 63 | 0   | 119| $6.5930 \times 10^{-5}$ | $2.2500 \times 10^{-6}$ | 10 | 52 | 70 | 133| $9.2583 \times 10^{-5}$ | $2.3373 \times 10^{-3}$ |
| 17  | 79 | 79 | 0   | 159| $8.5136 \times 10^{-5}$ | $4.5443 \times 10^{-6}$ | 24 | 56 | 55 | 136| $7.3576 \times 10^{-5}$ | $6.2449 \times 10^{-4}$ |
| 18  | 47 | 222| 16  | 285| $4.4363 \times 10^{-1}$ | $2.6364 \times 10^{-2}$ | 9 | 47 | 84 | 141| $9.0335 \times 10^{-5}$ | $2.4064 \times 10^{-3}$ |
| 19  | 54 | 89 | 0   | 144| $6.6094 \times 10^{-5}$ | $2.5192 \times 10^{-6}$ | 308| 126| 68 | 503| $6.8994 \times 10^{-5}$ | $1.0272 \times 10^{-3}$ |
| 20  | 97 | 98 | 0   | 196| $7.8812 \times 10^{-5}$ | $6.6541 \times 10^{-3}$ | 19 | 79 | 72 | 171| $9.1718 \times 10^{-5}$ | $6.4603 \times 10^{-3}$ |

Next, we compare the numerical performance of different noise type. For comparing the performances, we adopt the formula $Precision = |log_{10}(|f_k|)|$ and regard the NNE case as a benchmark for comparison. The cases of constant noise (CNE and CGNE) and exact form (NNE) are referred to in Figure 5. It is clear that the exact case has the best performance and Algorithm 1 can achieve a reasonable accuracy for the constant case. Meanwhile, the performance of CGNE case is better than that of CNE case. Similarly, Figure 6 reports the numerical performance for the vanishing cases (VNE and VGNE) and the exact form (NNE). From Figure 6, the performance of the VGNE case is comparable with that of the exact (NNE) case. Meanwhile, the performance of the vanishing error cases are generally better than that in the constant error cases.

Figure 5. Performance of Algorithm 1 for the NNE, CNE and CGNE cases.
5.3. Application to Some DC Problems

In this subsection, we test some unconstrained DC examples to illustrate the effectiveness of Algorithm 1. These examples come from [42–44]. Usually, the DC function has the form: \( \psi(x) = f(x) - g(x) \). If we take \( h(x) = -g(x) \), the problems are in the form of (1).

**Problem 1.** Dimension: \( N = 2 \),
Component functions: \( f(x) = |x_1 - 1| + 200 \max\{0, |x_1| - 2 \} \), \( g(x) = 100(|x_1| - x_2) \),
Relevant information: \( x_0 = (2, 5)^T, x^* = (1, 1)^T, \psi^* = 0 \).

**Problem 2.** Dimension: \( N = 4 \),
Component functions: \( f(x) = |x_1 - 1| + 200 \max\{0, |x_1| - 2 \} + 180 \max\{0, |x_3| - 4 \} + |x_3 - 1| + 10.1(|x_2 - 1| + |x_4 - 1|) + 4.95|x_2 + x_4 - 2| \), \( g(x) = 100(|x_1| - x_2) + 90(|x_3| - x_4) + 4.95|x_2 - x_4| \),
Relevant information: \( x_0 = (1, 3, 3, 1)^T, x^* = (1, 1, 1, 1)^T, \psi^* = 0 \).

**Problem 3.** Dimension: \( N = 2, 5, 10 \),
Component functions: \( f(x) = N \max\{|x_i| : i = 1, \ldots, N\} \), \( g(x) = \sum_{i=1}^{N}|x_i| \),
Relevant information: \( x_0 = \{i, i = 1, \ldots, [N/2]\}, -i, i = [N/2] + 1, \ldots, N \), \( x^* = (x^*_1, \ldots, x^*_N)^T, x^*_i = a \) or \( x^*_i = -a, a \in \mathbb{R}, i = 1, \ldots, N, \psi^* = 0 \).

**Problem 4.** Dimension: \( N = 4 \),
Component functions: \( f(x) = x_1^2 + (x_1 - 1)^2 + 2(x_1 - 2)^2 + (x_1 - 3)^2 + 2x_1^2 + (x_2 - 1)^2 + 2(x_2 - 2)^2 + x_2^2 + (x_3 - 1)^2 + 2(x_3 - 2)^2 + (x_3 - 3)^2 + 2x_3^2 + (x_4 - 1)^2 + 2(x_4 - 2)^2; \)
\( g(x) = \max\{(x_1 - 2)^2 + x_1^2, (x_3 - 2)^2 + x_3^2\} + \max\{(x_1 - 1)^2 + (x_2 - 2)^2, (x_3 - 2)^2 + (x_4 - 1)^2\} + \max\{(x_1 - 3)^2 + (x_2 - 2)^2 + x_2^2, (x_3 - 3)^2 + x_3^2\} + \max\{x_1^2 + (x_2 - 2)^2, x_3^2 + (x_4 - 2)^2\} + \max\{(x_1 - 1)^2 + (x_2 - 2)^2, (x_3 - 1)^2 + (x_4 - 2)^2\} \),
Relevant information: \( x_0 = (3, 1, 3, 1)^T, x^* = (7/3, 1/3, 0, 5, 2)^T, \psi^* = 11/6 \).

**Problem 5.** Dimension: \( N = 2, 5 \),
Component functions: \( f(x) = 20 \max\{|\sum_{i=1}^{N}(x_i - x^*_i)^2| : j = 1, 2, \ldots, 20\} \).
\[ g(x) = \sum_{j=1}^{20} |\sum_{i=1}^{N}(x_i - x_i^*)t_{ij}^{-1}|, \quad t_j = 0.05j, \quad j = 1, 2, \ldots, 20, \]

*Relevant information:* \( x^0 = (1/N, 0, \ldots, 0)^T, \quad x^* = (1/N, 1/N, \ldots, 1/N)^T, \quad \psi^* = 0. \)

**Problem 6. Dimension:** \( N = 2, 4, \)

*Component functions:* \( f(x) = \sum_{j=1}^{100} |\sum_{i=1}^{N}(x_i - x_i^*)t_{ij}^{-1}|, \quad t_j = 0.01j, \quad j = 1, 2, \ldots, 100, \)

\[ g(x) = \max\{ |\sum_{i=1}^{N}(x_i - x_i^*)t_{ij}^{-1}| : j = 1, 2, \ldots, 100 \} \]

*Relevant information:* \( x^0 = (0, 0, \ldots, 0)^T, \quad x^* = (1/N, 1/N, \ldots, 1/N)^T, \quad \psi^* = 0. \)

For the effectiveness of Algorithm 1, we compare it with the TCM algorithm, NCVX algorithm and penalty NCVX algorithm in [42]. The values of parameters in Algorithm 1 are that: \( m_1 = 0.01, \quad \kappa = 0.9, \quad \tau = 2, \quad \gamma = 2, \quad M_0 = 5, \quad R_0 = 10 \) and \( Tol = 10^{-3} \). The results can be seen in Table 11. Meanwhile the * in Table 11 means the obtained value is not optimal. From Table 11, we have that Algorithm 1 can successfully solve these DC problems, however the TCM algorithm cannot solve Problem 4, NCVX algorithm cannot solve Problems 1 and 4 and the penalty NCVX algorithm cannot solve Problem 1. Then, Algorithm 1 is reliable. From the obtained function value and the number of function evaluations, Algorithm 1 is also effective.

**Table 11.** The numerical results for Algorithm 1, TCM, NCVX and PNCVX algorithms in Problems 1–5.

| Pr | n | Algorithm 1 | TCM | NCVX | PNCVX |
|----|---|-------------|-----|------|-------|
|    |   | Nf | fk   | Nf | fk | Nf | fk | Nf | fk | Nf | fk | Nf | fk | Nf | fk |
| 1  | 2 | 204| 7.4219×10^{-10} | 526 | 8.453×10^{-9} | 20 | 1.0000 * | 48 | 1.0000 * | 0 |
| 2  | 4 | 69 | 6.4293×10^{-7}  | 517 | 3.7104×10^{-9} | 59 | 6.8183×10^{-6} | 59 | 5.2088×10^{-7} | 0 |
| 3  | 2 | 6  | 2.0000×10^{-11} | 125 | 4.0552×10^{-11} | 19 | 1.2800×10^{-8} | 4  | 1.2500×10^{-7} | 0 |
| 3  | 5 | 30 | 3.7100×10^{-6}  | 399 | 3.6698×10^{-10} | 25 | 2.5204×10^{-5} | 87 | 7.1731×10^{-8} | 0 |
| 3  | 10| 91 | 1.4116×10^{-4}  | 844 | 1.0639×10^{-9}  | 34 | 6.0636×10^{-8} | 78 | 2.2092×10^{-8} | 0 |
| 4  | 4 | 21 | 1.8333           | 307 | 9.2000 *         | 5  | 9.2000 *        | 34 | 1.8333           | 11/6 |
| 5  | 2 | 32 | 5.7727×10^{-6}  | 149 | 1.9922×10^{-10} | 19 | 1.1842×10^{-8} | 24 | 7.5230×10^{-9}  | 0 |
| 5  | 5 | 12 | 4.0654×10^{-7}  | 1174| 7.1262×10^{-9}  | 13 | 9.6573×10^{-9} | 12 | 1.1546×10^{-10} | 0 |

For the above DC problems, we consider the vanishing noise error (VNE) case and the exact (NNE) case. We also take 1000 as the upper limit of function evaluated. The algorithm stops when \( V_k \leq Tol \) holds or the number of function evaluated is over 1000. For the vanishing noise case, we set \( \theta_i = \min\{0.01, |x - x^*|/100\} \) and \( \epsilon_i = \min\{0.01, |x - x^*|/100\} \) except Problem 3. In Problem 3, the optimal solutions vary with the dimensions, then we set \( \theta_i = \min\{0.01, |x|^2/100\} \) and \( \epsilon_i = \min\{0.01, |x|^2/100\} \). Table 12 presents the results for the vanishing noise error case (VNE) and the exact case (NNE). The column \( Pr \) in Table 12 denotes the index of problems. We also compute the Precision. However it is not suitable since the optimal value is not 0. To deal with that, we take \( a_k = (f_k - f^*)/f^* \) and Precision = \( |\log_{10}|a_k|| \). The numerical results are reported as follows.

From Table 12, Algorithm 1 can successfully solve the above DC problems in a higher precision and is effective to the VNE case in a reasonable accuracy. Then, Algorithm 1 is effective and reliable to the above DC problems. During the numerical experiment, we also focus on the variation of parameters \( \eta \) and \( \mu \), which are both bounded and the parameter \( \eta \) eventually keeps unchanged, which can be illustrated in Figures 7 and 8.
Values of parameters $\eta$ and $\mu$ for example 3 with $n=5$ in VNE case

Figure 7. The values of $\eta$ and $\mu$ in Problem 3 with $N = 5$ in the VNE case.

Values of parameters $\eta$ and $\mu$ in Example 6 with $n=4$ for VNE case

Figure 8. The values of $\eta$ and $\mu$ in Problem 6 with $N = 4$ in the VNE case.
Table 12. The numerical results of Algorithm 1 for these DC problems in VNE and NNE cases.

| Pr | Dim | Algorithm 1 for VNE Case | Algorithm 1 for NNE Case |
|----|-----|--------------------------|--------------------------|
|    |     | $V_k$                     | $f_k$                     | $\text{Precision}$ | $Nu$ | $NS$ | $RS$ | $NF$ | $V_k$                     | $f_k$ | $\text{Precision}$ |
| 1  | 2   | $7.3309 \times 10^{-4}$   | $-9.8726 \times 10^{-3}$ | 2.0044            | 3    | 200  | 1    | 204  | $1.8221 \times 10^{-6}$ | $7.4219 \times 10^{-10}$ | 9.1295  |
| 2  | 4   | $7.6724 \times 10^{-4}$   | $1.7527 \times 10^{-6}$  | 5.7563            | 5    | 63   | 2    | 69   | $1.5569 \times 10^{-5}$ | $6.4293 \times 10^{-7}$ | 6.1918  |
| 3  | 2   | $6.4421 \times 10^{-5}$   | $-4.9703 \times 10^{-3}$ | 2.3010            | 0    | 5    | 0    | 6    | $3.2601 \times 10^{-8}$ | $2.0000 \times 10^{-11}$ | 10.6990 |
| 3  | 5   | $9.8576 \times 10^{-4}$   | $-8.7570 \times 10^{-3}$ | 2.0555            | 1    | 28   | 0    | 30   | $2.3067 \times 10^{-5}$ | $3.7122 \times 10^{-6}$ | 5.4304  |
| 3  | 4   | $2.6637 \times 10^{-4}$   | $1.480 \times 10^{-7}$   | 0.9401            | 1    | 89   | 0    | 91   | $2.9760 \times 10^{-4}$ | $1.4116 \times 10^{-4}$ | 3.8503  |
| 4  | 2   | $8.2023 \times 10^{-4}$   | $1.8234$                  | 2.2661            | 1    | 19   | 0    | 21   | $8.8458 \times 10^{-4}$ | $1.8333$ | 4.7404  |
| 5  | 2   | $6.2198 \times 10^{-4}$   | $1.6864 \times 10^{-4}$  | 3.7730            | 6    | 25   | 0    | 32   | $9.2682 \times 10^{-4}$ | $5.7727 \times 10^{-6}$ | 5.2386  |
| 5  | 5   | $8.0435 \times 10^{-4}$   | $-6.5385 \times 10^{-4}$ | 3.1845            | 2    | 9    | 0    | 12   | $7.5698 \times 10^{-4}$ | $4.0654 \times 10^{-7}$ | 6.3909  |
| 6  | 2   | $1.1607 \times 10^{-4}$   | $1.8586 \times 10^{-5}$  | 4.7308            | 2    | 4    | 2    | 7    | $1.0840 \times 10^{-12}$ | $5.3857 \times 10^{-15}$ | 14.2688 |
| 6  | 4   | $1.3749 \times 10^{-4}$   | $-2.6406 \times 10^{-3}$ | 2.5850            | 4    | 11   | 2    | 16   | $1.3123 \times 10^{-4}$ | $1.9301 \times 10^{-7}$ | 6.7144  |
6. Conclusions

In this paper, we consider a special class of nonconvex and nonsmooth composite problem. The problem is constituted by the sum of two functions, one is finite convex with inexact information and the other is a nonconvex function (lower-$C^2$). For the nonconvex function, we utilize the convexification technique and adjust the parameter dynamically to make sure the linearization errors of the augment function nonnegative and construct the corresponding cutting plane models. Then, we regard the sum of the convex function and the augment function as an approximate function. For the convex function with inexact information, we construct the cutting plane model by its inexact information and notice that the cutting plane model may not be below the convex function. Then, the sum of the cutting plane models of the convex function with inexact information and the augment function is regarded as a cutting plane model of the approximate function. After that, we design an adaptive proximal bundle method. Meanwhile, for the convex function with inexact information, we utilize the noise management strategy and update adaptively the proximal parameter to reduce the influence of inexact information. Two polynomial functions including five different inexact types and six DC problems with different dimension are referred to in the numerical experiment. The preliminary numerical results show our algorithm is interesting and reliable. Meanwhile, our method can also be applied to some constraint problems and stochastic programming in the future.

Author Contributions: Conceptualization, X.W. and L.P.; methodology, X.W.; software, X.W., Q.W. and M.Z.; validation, X.W., L.P. and M.Z.; formal analysis, X.W. and Q.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data can be found in the manuscript.

Acknowledgments: We are greatly indebted to three anonymous referees for many helpful comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Condat, L. A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. *J. Optim. Theory Appl.* 2013, 158, 460–479. [CrossRef]
2. Li, G.Y.; Pong, T.K. Global convergence of splitting methods for nonconvex composite optimization. *SIAM J. Optim.* 2015, 25, 2434–2460. [CrossRef]
3. Hong, M.; Luo, Z.Q. On the linear convergence of the alternating direction method of multipliers. *Math. Program.* 2017, 162, 165–199. [CrossRef]
4. Li, D.; Pang, L.P.; Cher, S. A proximal alternating linearization method for nonconvex optimization problems. *Optim. Method Softw.* 2014, 29, 771–785. [CrossRef]
5. Burke, J.V.; Lewis, A.S.; Overton, M.L. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Optim.* 2005, 15, 751–779. [CrossRef]
6. Kiwiel, K.C. A method of centers with approximate subgradient linearizations for nonsmooth convex optimization. *SIAM J. Optim.* 2008, 18, 1467–1489. [CrossRef]
7. Yuan, G.L.; Meng, Z.H.; Li, Y. A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimizations and nonlinear equations. *J. Optim. Theory Appl.* 2016, 168, 129–152. [CrossRef]
8. Yuan, G.L.; Sheng, Z. *Nonsmooth Optimization Algorithms*; Science Press: Beijing, China, 2017.
9. Yuan, G.L.; Wei, Z.X.; Li, G. A modified Polak-Ribiére-Polyak conjugate gradient algorithm for nonsmooth convex programs. *J. Comput. Appl. Math.* 2014, 255, 86–96. [CrossRef]
10. Lv, J.; Pang, L.P.; Meng, F.F. A proximal bundle method for constrained nonsmooth nonconvex optimization with inexact information. *J. Glob. Optim.* 2015, 70, 517–549. [CrossRef]
11. Yang, Y.; Pang, L.P.; Ma, X.F.; Shen, J. Constrained nonconvex nonsmooth optimization via proximal bundle method. *J. Optim. Theory Appl.* 2014, 163, 900–925. [CrossRef]
12. Fuduli, A.; Gaudioso, M.; Giallombardo, G. Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. *SIAM J. Optim.* 2004, 14, 743–756. [CrossRef]
