algorithms

Article

A Generalized Alternating Linearization Bundle Method for Structured Convex Optimization with Inexact First-Order Oracles

Chunming Tang *, Yanni Li, Xiaoxia Dong and Bo He

College of Mathematics and Information Science, Guangxi University, Nanning 540004, China; ynli@st.gxu.edu.cn (Y.L.); xxdong@st.gxu.edu.cn (X.D.); bohe@st.gxu.edu.cn (B.H.)

* Correspondence: cmtang@gxu.edu.cn

Received: 21 January 2020; Accepted: 9 April 2020; Published: 14 April 2020

Abstract: In this paper, we consider a class of structured optimization problems whose objective function is the summation of two convex functions: f and h, which are not necessarily differentiable. We focus particularly on the case where the function f is general and its exact first-order information (function value and subgradient) may be difficult to obtain, while the function h is relatively simple. We propose a generalized alternating linearization bundle method for solving this class of problems, which can handle inexact first-order information of on-demand accuracy. The inexact information can be very general, which covers various oracles, such as inexact, partially inexact and asymptotically exact oracles, and so forth. At each iteration, the algorithm solves two interrelated subproblems: one aims to find the proximal point of the polyhedron model of f plus the linearization of h; the other aims to find the proximal point of the linearization of f plus h. We establish global convergence of the algorithm under different types of inexactness. Finally, some preliminary numerical results on a set of two-stage stochastic linear programming problems show that our method is very encouraging.

Keywords: nonsmooth convex optimization; bundle method; alternating linearization; on-demand accurary; global convergence

1. Introduction

In this paper, we consider the following structured convex optimization problem

$$F_* := \min_{x \in \mathbb{R}^n} \{F(x) := f(x) + h(x)\}, \tag{1}$$

where $f:\ \mathrm{dom}\, h \to \mathbb{R}$ and $h:\ \mathbb{R}^n \to (-\infty, \infty]$ are closed proper convex functions, but not necessarily differentiable, and $\mathrm{dom}\, h := \{x :\ h(x) < \infty\}$ is the effective domain of h. Problems of this type frequently arise in practice, such as compressed sensing [1], image reconstruction [2], machine learning [3], optimal control [4] and power system [5–8], and so forth. The following are three interesting examples.

Example 1. *(ℓ_1 minimization in compressed sensing). The signal recovery problems in compressed sensing [1] usually take the following form*

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|Ax - b\|^2 + \lambda \|x\|_1, \tag{2}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $\lambda > 0$ and $\|x\|_1 := \sum_{i=1}^n |x_i|$, which aims to get a sparse solution x of the linear system $Ax = b$. Note that by defining $f(x) = \frac{1}{2}\|Ax - b\|^2$ and $h(x) = \lambda\|x\|_1$, (2) is of the form of (1).

Example 2. *(Regularized risk minimization). At the core of many machine learning problems is to minimize a regularized risk function [9,10]:*

$$\min_{x\in\mathbb{R}^n} R_{\text{emp}}(x) + \lambda\Omega(x), \tag{3}$$

where $R_{\text{emp}}(x) := \frac{1}{m}\sum_{i=1}^{m} l(u_i, v_i, x)$ is the empirical risk, $\{(u_i, v_i),\ i = 1, \cdots, m\}$ is a training set, and l is a convex loss function measuring the gap between v and the predicted values generated by using x. In general, $R_{\text{emp}}(x)$ is a nondifferentiable and computationally expensive convex function, whereas the regularization term $\Omega(x)$ is a simple convex function, say $\Omega(x) = \frac{1}{2}\|x\|_2^2$ or $\Omega(x) = \|x\|_1$. By defining $f(x) = R_{\text{emp}}(x)$ and $h(x) = \lambda\Omega(x)$, (3) is also of the form of (1).

Example 3. *(Unconstrained transformation of a constrained problem). Given a constrained problem*

$$\min\ \{f(x) :\ x \in X\}, \tag{4}$$

where f is a convex function and $X \subseteq \mathbb{R}^n$ is a convex set. By introducing the indicator function δ_X of X, that is, $\delta_X(x)$ equals 0 on X and infinity elsewhere, problem (4) can be written equivalently as

$$\min_{x\in\mathbb{R}^n} f(x) + \delta_X(x). \tag{5}$$

Clearly, by setting $h(x) = \delta_X(x)$, (5) becomes the form of (1). We note that such transformation could be very efficient in practice if the set X has some special structure [11,12].

The design of methods for solving problems of the form (1) has attracted the attention of many researchers. We mention here four classes of these methods—operator splitting methods [13–15], alternating direction methods of multipliers [5,16–19], alternating linearization methods [20,21], and alternating linearization bundle method [22]. They all fall within the well-known class of first-order black-box methods, that is, it is assumed that there is an *oracle* that can return the (approximate) function value and one arbitrary (approximate) subgradient at any given point. Regarding the above methods, we are concerned about the following three points:

- Smoothness of one or both functions in the objective has been assumed for many of the methods.
- Except for Reference [22], they all require the exact computation of the function values and (sub)gradients.
- The alternating linearization methods [20,21] essentially assume that both functions f and h are "simple" in the sense that minimizing the function plus a separable convex quadratic function is easy.

However, for some practical problems, the functions may be nondifferentiable (nonsmooth), not easy to handle, and computationally expensive in the sense that the exact first-order information may be impossible to calculate, or be computable but difficult to obtain. For example, if f has the form

$$f(x) := \sup\{\phi_u(x) :\ u \in U\},$$

where U is an infinite set and $\phi_u(x) :\ \mathbb{R}^n \to \mathbb{R}$ is convex for any $u \in U$, then it is often difficult to calculate the exact function value $f(x)$. But for any tolerance $\epsilon > 0$, we may usually find a lower approximation $f_x^\epsilon \approx f(x)$ in finite time such that $f_x^\epsilon \in [f(x) - \epsilon, f(x)]$ and $f_x^\epsilon = \phi_{u_\epsilon}(x)$ for some $u_\epsilon \in U$. Then we can take a subgradient of ϕ_{u_ϵ} at x as an approximate subgradient of f at x. Another example is two-stage stochastic programming (see, e.g., References [23,24]), in which the function value is generated after solving a series of linear programs (details will be given in the section of numerical experiments), therefore its accuracy is determined by the tolerance of the linear programming solver. Some other practical examples, such as Lagrangian relaxation, chance-constrained programs and convex composite functions, can be found in Reference [25].

Based on the above observation, in this paper, we focus particularly on the case where the function f is general, possibly nonsmooth and its exact function values and subgradients may be difficult to obtain, whereas the function h is assumed to be relatively simple. Our main goal is to provide an efficient method, namely, the improved alternating linearization bundle method, for such kind of structured convex optimization problems. The basic tool we used here to handle nonsmoothness and inexactness is the bundle technique, since in the nonsmooth optimization community, bundle methods [26–29] are regarded as the most robust and reliable methods, whose variants have been well studied for handling inexact oracles [23,25,30–33].

Roughly speaking, our method generalizes the alternating linearization bundle method of Kiwiel [22] from exact and inexact oracles to various oracles, including exact, inexact, partially inexact, asymptotically exact and partially asymptotically exact oracles. These oracles are covered by the so-called *on-demand accuracy oracles* proposed by de Oliveira and Sagastizábal [23], whose accuracy is controlled by two parameters: a descent target and an error bound. More precisely, the accuracy is bounded by an error bound whenever the function estimation reaches a certain descent target. The most advantage of oracles with on-demand accuracy is that the function and subgradient estimations can be rough without an accuracy control for some "non-critical" iterates, thus the computational effort can be saved.

At each iteration, the proposed algorithm alternately solves two interrelated subproblems: one is to find the proximal point of the polyhedron model of f plus the linearization of h; the other is to find the proximal point of the linearization of f plus h. We establish global convergence of the algorithm under different types of inexactness. Finally, some preliminary numerical results on a set of two-stage stochastic linear programming problems show that our method is very encouraging.

This paper is organized as follows. In Section 2, we recall the condition of the inexact frist-order oracles. In Section 3, we present an improved alternating linearization bundle method for structured convex optimization with inexact first-order oracles and show some properties of the algorithm. In Section 4, we establish global convergence of the algorithm under different types of inexactness. In Section 5, we provide some numerical experiments on two-stage stochastic linear programming problems. The notations are standard. The Euclidean inner product in $\mathbb{R}^n$ is denoted by $\langle x, y\rangle := x^T y$, and the associated norm by $\|\cdot\|$.

2. Preliminaries

For a given constant $\epsilon \geq 0$, the ϵ-subdifferential of function f at x is defined by (see Reference [34])

$$\partial_\epsilon f(x) := \{g \in \mathbb{R}^n : \ f(y) \geq f(x) + \langle g, y - x\rangle - \epsilon, \ \forall y \in \mathbb{R}^n\},$$

with $\partial f(x) := \partial_0 f(x)$ being the usual subdifferential in convex analysis [35]. Each element $g \in \partial f(x)$ is called a subgradient. For simplicity, we use the following notations:

f_x: the approximate f value at x, that is, $f_x \approx f(x)$;

g_x: an approximate subgradient of f at x, that is, $g_x \approx g(x) \in \partial f(x)$;

F_x: the approximate F value at x, that is, $F_x := f_x + h(x)$.

Aiming at the special structure of problem (1), we present a slight variant of the oracles with on-demand accuracy proposed in Reference [23] as follows: for a given $x \in \mathbb{R}^n$, a descent target γ_x and an error bound $\varepsilon_x \geq 0$, the approximate values f_x, g_x and F_x satisfy the following condition

$$\begin{cases} f_x = f(x) - \eta(\gamma_x) \ \text{ with unknown } \eta(\gamma_x) \geq 0, \\ g_x \in \partial_{\eta(\gamma_x)} f(x), \text{ and} \\ \text{whenever } \ F_x \leq \gamma_x \ (\textit{descent target reached}), \text{ the relation } \eta(\gamma_x) \leq \varepsilon_x \text{ holds.} \end{cases} \tag{6}$$

From the relations in (6), we see that although the error $\eta(\gamma_x)$ is unknown, it has to be restricted within the error bound ε_x whenever the descent target $F_x \leq \gamma_x$ is reached. This ensures that the exact and inexact function values satisfy

$$f_x \in [f(x) - \varepsilon_x, f(x)] \ \text{ and } \ f(x) \in [f_x, f_x + \varepsilon_x], \quad \text{whenever } \ F_x \leq \gamma_x. \tag{7}$$

The advantages of oracle (6) are that: (1) if the descent target is not reached, the calculation of oracle information can be rough without an accuracy control, which can potentially reduce the computational cost; (2) by properly choosing the parameters γ_x and ε_x, the oracle (6) covers various existing oracles:

- Exact (Ex) [12,21]: set $\gamma_x = +\infty$ and $\varepsilon_x = 0$;
- Partially Inexact (PI) [24]: set $\gamma_x < +\infty$ and $\varepsilon_x = 0$;
- Inexact (IE) [11,25,32,36,37]: set $\gamma_x = +\infty$ and $\varepsilon_x \equiv \varepsilon > 0$ (possibly unknown);
- Asymptotically Exact (AE) [38,39]: set $\gamma_x = +\infty$ and $\varepsilon_x \to 0$ along the iterative process;
- Partially Asymptotically Exact (PAE) [23]: set $\gamma_x < +\infty$ and $\varepsilon_x \to 0$.

3. The Generalized Alternating Linearization Bundle Method

In this section, we present our generalized alternating linearization bundle method with inexact first-order oracles for solving (1).

Let k be the current iteration index, x^j, $j \in J^k \subseteq \{1, \cdots, k\}$ be given points generated by previous iterations, and the corresponding approximate values f_{x^j}/g_{x^j} be produced by the oracle (6). For notational convenience, we denote

$$f_x^j := f_{x^j},\ g_x^j := g_{x^j},\ F_x^j := F_{x^j},\ \varepsilon_x^j := \varepsilon_{x^j},\ \gamma_x^j := \gamma_{x^j}.$$

The approximate linearizations of f at x^j are given by

$$f_j(\cdot) := f_x^j + \langle g_x^j, \cdot - x^j \rangle,\ \ j \in J^k.$$

From the second relation in (6), we have

$$f(\cdot) \geq f(x^j) + \langle g_x^j, \cdot - x^j \rangle - \eta(\gamma_x^j) = f_j(\cdot),$$

which implies that f_j is a lower approximation to f. Next, it is natural to define the polyhedral inexact cutting-plane model of f by

$$\check{f}_k(\cdot) := \max_{j \in J_k} \{f_j(\cdot)\}, \tag{8}$$

which is obviously a lower polyhedral model for f, that is, $\check{f}_k(\cdot) \leq f(\cdot)$.

Let $\hat{x}^k$ (called *stability center*) be the "best" point obtained so far, which satisfies that $\hat{x}^k = x^{k(l)}$ for some $k(l) \leq k$. Frequently, it holds that $f_x^{k(l)} = \min\limits_{j=1,\cdots,k} f_x^j$. Thus, from (7), we have

$$f(\hat{x}^k) \in [f_{\hat{x}}^k, f_{\hat{x}}^k + \varepsilon_x^{k(l)}], \quad \text{whenever } \ F_{\hat{x}}^k \leq \gamma_{\hat{x}}^k. \tag{9}$$

By applying the bundle idea to the "complex" function f, and keeping the simple function h unchanged, similar to traditional proximal bundle methods (see, e.g., Reference [28]), we may solve the following subproblem to obtain a new iterate x^{k+1}:

$$x^{k+1} := \arg\min\ \check{f}_k(\cdot) + h(\cdot) + \frac{1}{2t_k}\|\cdot - \hat{x}^k\|^2, \tag{10}$$

where $t_k > 0$ is a proximal parameter. However, subproblem (10) is generally not easy to solve, so by making use of the alternating linearization idea of Kiwiel [22], we solve two easier subproblems instead of (10). These two subproblems are interrelated: one is to find the proximal point of the polyhedron model $\check{f}$ plus the linearization of h, aiming at generating an aggregate linear model of f for use in the second subproblem; the other is to find the proximal point of the aggregate linear model of f plus h, aiming at obtaining a new trial point.

Now, we are ready to present the details of our algorithm, which generalizes the work of Kiwiel [22]. We note that the choice of the model function $\check{f}_k$ in the algorithm may be different from the form of (8), since the subgradient aggregation strategy [40] is used to compress the bundle. The algorithm generates three sequences of iterates as follows: $\{y^k\}$, the sequence of intermediate points, at which the aggregate linear models of f are generated; $\{x^k\}$, the sequence of trial points; $\{\hat{x}^k\}$, the sequence of stability centers.

We make some comments about Algorithm 1 as follows.

Algorithm 1 Generalized alternating linearization bundle method

Step 0 (Initialization). Select an initial point $x^1 \in \mathbb{R}^n$, constants $\kappa \in (0,1)$, $t_{\min} > 0$, and an initial stepsize $t_1 \geq t_{\min}$. Call the oracle (6) at x^1 to compute the approximate values f_x^1 and g_x^1. Choose an initial error bound $\varepsilon_x^1 \geq 0$ and a descent target $\gamma_x^1 = +\infty$. Set $\hat{x}^1 := x^1$, $f_{\hat{x}}^1 := f_x^1$, $F_{\hat{x}}^1 := f_{\hat{x}}^1 + h(\hat{x}^1)$, $\bar{f}_0 := f_1$, and $\bar{h}_0(\cdot) := h(x^1) + \langle p_h^0, \cdot - x^1 \rangle$ with $p_h^0 \in \partial h(x^1)$. Let $i_t^1 := 0$, $l := 1$, $k(l) := 1$ and $k := 1$.

Step 1 (Model selection). Choose $\check{f}_k : \mathbb{R}^n \to \mathbb{R}$ closed convex and such that

$$\max\{\bar{f}_{k-1}, f_k\} \leq \check{f}_k \leq f.$$

Step 2 (Solve f-subproblem). Set

$$y^{k+1} := \arg\min \left\{ \phi_f^k(\cdot) := \check{f}_k(\cdot) + \bar{h}_{k-1}(\cdot) + \frac{1}{2t_k} \| \cdot - \hat{x}^k \|^2 \right\}, \tag{11}$$

$$\bar{f}_k(\cdot) := \check{f}_k(y^{k+1}) + \langle p_f^k, \cdot - y^{k+1} \rangle \;\text{ with }\; p_f^k := \frac{1}{t_k}(\hat{x}^k - y^{k+1}) - p_h^{k-1}. \tag{12}$$

Step 3 (Solve h-subproblem). Set

$$x^{k+1} := \arg\min \left\{ \phi_h^k(\cdot) := \bar{f}_k(\cdot) + h(\cdot) + \frac{1}{2t_k} \| \cdot - \hat{x}^k \|^2 \right\}, \tag{13}$$

$$\bar{h}_k(\cdot) := h(x^{k+1}) + \langle p_h^k, \cdot - x^{k+1} \rangle \;\text{ with }\; p_h^k := \frac{1}{t_k}(\hat{x}^k - x^{k+1}) - p_f^k. \tag{14}$$

Step 4 (Stopping criterion). Compute

$$v_k := F_{\hat{x}}^k - \left[\bar{f}_k(x^{k+1}) + h(x^{k+1}) \right], \; p^k := \frac{1}{t_k}(\hat{x}^k - x^{k+1}), \; \epsilon_k := v_k - t_k \|p^k\|^2. \tag{15}$$

If $\max\{\|p^k\|, \epsilon_k\} = 0$, stop.

Step 5 (Noise attenuation). If $v_k < -\epsilon_k$, set $t_k := 10t_k$, $i_t^k := k$, and go back to Step 2.

Step 6 (Call oracle). Select a new error bound $\varepsilon_x^{k+1} \geq 0$ and a new descent target $\gamma_x^{k+1} \in \mathbb{R} \cup \{+\infty\}$. Call the oracle (6) to compute f_x^{k+1} and g_x^{k+1}.

Step 7 (Descent test). If the descent condition

$$F_x^{k+1} \leq F_{\hat{x}}^k - \kappa v_k \tag{16}$$

holds, set $\hat{x}^{k+1} := x^{k+1}$, $F_{\hat{x}}^{k+1} := F_x^{k+1}$, $i_t^{k+1} := 0$, $k(l+1) := k+1$, and $l := l+1$ (*descent step*); otherwise, set $\hat{x}^{k+1} := \hat{x}^k$, $F_{\hat{x}}^{k+1} := F_{\hat{x}}^k$, $i_t^{k+1} := i_t^k$, and $k(l+1) = k(l)$ (*null step*).

Step 8 (Stepsize updating). For a descent step, select $t_{k+1} \geq t_k$. For a null step, either set $t_{k+1} := t_k$ or choose $t_{k+1} \in [t_{\min}, t_k]$ if $i_t^{k+1} = 0$.

Step 9 (Loop). Let $k := k+1$, and go to Step 1.

Remark 1. *(i) Theoretically speaking, the model function $\check{f}_k$ can be the simplest form* $\max\{\bar{f}_{k-1}, f_k\}$, *but in order to keep numerical stability, it may additionally consist of some active linearizations.*

(ii) Alternately solving subproblems (11) and (13) can be regarded as the proximal alternating linearization method (e.g., Reference [21]) being applied to the function $\check{f}_k + h$*.*

(iii) If $\check{f}_k$ *is a polyhedral function, then subproblem (11) is equivalent to a convex quadratic programming and thus can be solved efficiently. In addition, if h is simple, subproblem (13) can also be solved easily, or even has a closed-form solution (say* $h(x) = \frac{1}{2}\|x\|^2$*).*

(iv) The role of Step 5 is to reduce the impact of inexactness. The algorithm loops between steps 2–5 by increasing the step size t_k *until* $v_k \geq -\varepsilon_k$*.*

(v) The stability center, descent target and error bound keep unchanged in the loop between Steps 2 and 5

(vi) In order to establish global convergence of the algorithm, the descent target and error bound at Step 6 should be suitably updated. Some detailed rules are presented in the next section.

The following lemma summarizes some fundamental properties of Algorithm 1, whose proof is a slight modification of that in [22], Lemma 2.2.

Lemma 1. *(i) The vectors* p_f^k *and* p_h^k *of* (12) *and* (14) *satisfy*

$$p_f^k \in \partial \check{f}_k(y^{k+1}) \text{ and } p_h^k \in \partial h(x^{k+1}). \tag{17}$$

The linearizations $\bar{f}_k$*,* $\bar{h}_k$*,* $\bar{F}_k$ *satisfy the following inequalities*

$$\bar{f}_k \leq \check{f}_k, \ \ \bar{h}_k \leq h \text{ and } \bar{F}_k := \bar{f}_k + \bar{h}_k \leq F. \tag{18}$$

(ii) The aggregate subgradient p_k *defined in* (15) *and the above linearization* $\bar{F}_k$ *can be expressed as follows*

$$p^k = p_f^k + p_h^k = \frac{1}{t_k}(\hat{x}^k - x^{k+1}), \tag{19}$$

$$\bar{F}_k(\cdot) = \bar{F}_k(x^{k+1}) + \langle p^k, \cdot - x^{k+1}\rangle.$$

(iii) The predicted descent v_k *and the aggregate linearization error* ϵ_k *of* (15) *satisfy*

$$v_k = t_k\|p^k\|^2 + \epsilon_k \text{ and } \epsilon_k = F_{\hat{x}}^k - \bar{F}_k(\hat{x}^k). \tag{20}$$

(iv) The aggregate linearization $\bar{F}_k$ *is also expressed*

$$F_{\hat{x}}^k - \epsilon_k + \langle p^k, \cdot - \hat{x}^k\rangle = \bar{F}_k(\cdot) \leq F(\cdot). \tag{21}$$

(v) Denote the optimality measure by

$$V_k := \max\{\|p^k\|, \epsilon_k + \langle p^k, \hat{x}^k\rangle\}, \tag{22}$$

which satisfies

$$V_k \leq \max\{\|p^k\|, \epsilon_k\}(1 + \|\hat{x}^k\|) \tag{23}$$

and

$$F_{\hat{x}}^k \leq F(x) + V_k(1 + \|x\|), \ \forall\, x. \tag{24}$$

(vi) *We have the relations*

$$v_k \geq -\epsilon_k \Leftrightarrow t_k\|p^k\|^2/2 \geq -\epsilon_k \Leftrightarrow v_k \geq t_k\|p^k\|^2/2, \ \ v_k \geq \epsilon_k. \tag{25}$$

Moreover, if $F_{\hat{x}}^k \leq \gamma_{\hat{x}}^k$, then we have $-\epsilon_k \leq \varepsilon_x^{k(l)}$ and

$$v_k \geq \max\left\{\frac{t_k\|p^k\|^2}{2}, |\epsilon_k|\right\} \ \ if\ v_k \geq -\epsilon_k, \tag{26}$$

$$V_k \leq \max\left\{\left(\frac{2v_k}{t_k}\right)^{1/2}, v_k\right\}(1+\|\hat{x}^k\|) \ \ if\ v_k \geq -\epsilon_k, \tag{27}$$

$$V_k < \left(\frac{2\varepsilon_x^{k(l)}}{t_k}\right)^{1/2}(1+\|\hat{x}^k\|) \ \ if\ v_k < -\epsilon_k. \tag{28}$$

Proof. (i) From the optimality condition of subproblem (11), we obtain

$$0 \in \partial\phi_f^k(y^{k+1}) = \partial\check{f}_k(y^{k+1}) + p_h^{k-1} + \frac{1}{t_k}(y^{k+1} - \hat{x}^k) = \partial\check{f}_k(y^{k+1}) - p_f^k,$$

which implies $p_f^k \in \partial\check{f}_k(y^{k+1})$. In addition, the fact that $\bar{f}_k(y^{k+1}) = \check{f}_k(y^{k+1})$ yields $\bar{f}_k \leq \check{f}_k$. Similarly, by the optimality condition of (14), we have

$$0 \in \partial\phi_h^k(x^{k+1}) = p_f^k + \partial h(x^{k+1}) + \frac{1}{t_k}(x^{k+1} - \hat{x}^k) = \partial h(x^{k+1}) - p_h^k,$$

which shows $p_h^k \in \partial h(x^{k+1})$. Further from $\bar{h}(x^{k+1}) = h(x^{k+1})$, we obtain $\bar{h}_k \leq h$. Finally, it follows that

$$\bar{F}_k = \bar{f}_k + \bar{h}_k \leq \check{f}_k + h \leq F.$$

(ii) By (14), we obtain

$$p_f^k + p_h^k = p_f^k + \frac{1}{t_k}(\hat{x}^k - x^{k+1}) - p_f^k = \frac{1}{t_k}(\hat{x}^k - x^{k+1}) = p^k.$$

Utilizing the linearity of $\bar{F}_k(\cdot)$, (12) and (19), we derive

$$\begin{aligned}
\bar{F}_k(\cdot) &= \bar{f}_k(\cdot) + \bar{h}_k(\cdot)\\
&= \check{f}_k(y^{k+1}) + \langle p_f^k, \cdot - y^{k+1}\rangle + h(x^{k+1}) + \langle p_h^k, \cdot - x^{k+1}\rangle\\
&= \bar{f}_k(x^{k+1}) - \langle p_f^k, x^{k+1} - y^{k+1}\rangle + \langle p_f^k, \cdot - y^{k+1}\rangle + h(x^{k+1}) + \langle p_h^k, \cdot - x^{k+1}\rangle\\
&= \bar{f}_k(x^{k+1}) + \langle p_f^k, \cdot - x^{k+1}\rangle + h(x^{k+1}) + \langle p_h^k, \cdot - x^{k+1}\rangle\\
&= \bar{F}_k(x^{k+1}) + \langle p^k, \cdot - x^{k+1}\rangle.
\end{aligned}$$

(iii) We obtain directly $v_k = \epsilon_k + t_k\|p^k\|^2$ from (15). Combining (15) and (ii), we have

$$\begin{aligned}
\epsilon_k &= v_k - t_k\|p^k\|^2\\
&= F_{\hat{x}}^k - [\bar{f}_k(x^{k+1}) + h(x^{k+1})] - t_k\|p^k\|^2\\
&= F_{\hat{x}}^k - \bar{F}_k(\hat{x}^k) + \langle p^k, \hat{x}^k - x^{k+1}\rangle - t_k\|p^k\|^2\\
&= F_{\hat{x}}^k - \bar{F}_k(\hat{x}^k).
\end{aligned}$$

(iv) Since $\epsilon_k = v_k - t_k\|p^k\|^2 = F_{\hat{x}}^k - [\bar{f}_k(x^{k+1}) + h(x^{k+1})] - t_k\|p^k\|^2$, the aggregate lineaization $\bar{F}_k(\cdot)$ satisfies

$$F_{\hat{x}}^k - \epsilon_k + \langle p^k, \cdot - \hat{x}^k\rangle = \bar{F}_k(x^{k+1}) + \langle p^k, \cdot - x^{k+1}\rangle = \bar{F}_k(\cdot) \leq F(\cdot).$$

(v) Using the Cauchy-Schwarz inequality in the definition (22) gives

$$\begin{aligned} V_k &= \max\{\|p_k\|, \epsilon_k + \langle p^k, \hat{x}^k\rangle\} \\ &\le \max\{\|p^k\|, \epsilon_k + \|p^k\|\|\hat{x}^k\|\} \\ &\le \max\{\|p^k\|, \epsilon_k\} + \|p^k\|\|\hat{x}^k\| \\ &\le \max\{\|p^k\|, \epsilon_k\}(1 + \|\hat{x}^k\|). \end{aligned}$$

From (21), we have

$$\begin{aligned} F_{\hat{x}}^k &\le F(x) + \epsilon_k - \langle p^k, x - \hat{x}^k\rangle \\ &= F(x) + \epsilon_k - \langle p^k, x\rangle + \langle p^k, \hat{x}^k\rangle \\ &\le F(x) + \|p^k\|\|x\| + \epsilon_k + \langle p^k, \hat{x}^k\rangle \\ &\le F(x) + \max\{\|p^k\|, \epsilon_k + \langle p^k, \hat{x}^k\rangle\}(1 + \|x\|) \\ &= F(x) + V_k(1 + \|x\|), \ \ \forall\, x. \end{aligned}$$

(vi) By (iii), it is easy to get (25). Next, by (18), (20) and (9), we conclude that, if $F_{\hat{x}}^k \le \gamma_{\hat{x}}^k$,

$$-\epsilon_k = \bar{F}_k(\hat{x}^k) - F_{\hat{x}}^k \le F(\hat{x}^k) - F_{\hat{x}}^k = f(\hat{x}^k) - f_{\hat{x}}^k \le \varepsilon_x^{k(l)}.$$

Relation (26) follows from the facts that $v_k \ge \epsilon_k$ and $v_k \ge t_k\|p^k\|^2/2$. Relation (27) follows from (23), $\|p^k\| \le (\frac{2v_k}{t_k})^{1/2}$ and $\epsilon_k \le v_k$. Finally, if $v_k < -\epsilon_k$, we obtain $\|p^k\|^2 < \frac{-2\epsilon_k}{t_k}$, which together with $-\epsilon_k \le \varepsilon_x^{k(l)}$ shows that $\|p^k\| < (\frac{2\varepsilon_x^{k(l)}}{t_k})^{1/2}$, and therefore (28) holds. □

Remark 2. *Relation (17) shows that p_f^k is a subgradient of the model function $\check{f}_k$ at y^{k+1} and that p_h^k is a subgradient of h at x^{k+1}. V_k defined by (22) can be viewed as an optimality measure of the iterates, which will be proved to converge to zero in the next section. Relation (24) is also a test for optimality, in that $\hat{x}^k$ is an approximate optimal solution to problem (1) whenever V_k is sufficiently small.*

4. Global Convergence

This section aims to establish the global convergence of Algorithm 1 for various oracles. These oracles are controlled by two parameters: the error bound ε_x and the descent target γ_x. In Table 1, we present the choices of these two parameters for different type of instances described in Section 2, including Exact (Ex), Partially Inexact (PI), Inexact (IE), Asymptotically Exact (AE) and Partially Asymptotically Exact (PAE) oracles, where the constants are selected as θ, $\kappa \in (0, 1)$, and $\kappa_\epsilon \in (0, \kappa)$.

Table 1. The choices for the error bound and the descent target.

Instances	ε_x^{k+1}	γ_x^{k+1}
Ex	0	$+\infty$
PI	0	$F_{\hat{x}}^k - \theta\kappa v_k$
IE	$\varepsilon > 0$	$+\infty$
AE	$\kappa_\epsilon v_k$	$+\infty$
PAE	$\kappa_\epsilon v_k$	$F_{\hat{x}}^k - \theta\kappa v_k$

The following lemma is crucial to guarantee the global convergence of Algorithm 1.

Lemma 2. *The descent target is always reached at the stability centers, that is, $F_{\hat{x}}^k \le \gamma_{\hat{x}}^k$ for all $k \ge 1$.*

Proof. For instances Ex, IE and AE, since $\gamma_x^k = +\infty$, the claim holds immediately.

For instances PI and PAE, we have $\gamma_x^{k+1} = F_{\hat{x}}^k - \theta\kappa v_k$. Thus, for $k = 1$, from Step 0 it follows that $\hat{x}^1 = x^1$, $f_{\hat{x}}^1 = f_x^1$ and $\gamma_{\hat{x}}^1 = \gamma_x^1 = +\infty$. This implies $F_{\hat{x}}^1 = f_{\hat{x}}^1 + h(\hat{x}^1) \le \gamma_{\hat{x}}^1$. In addition, for $k \ge 2$, since $\theta \in (0,1)$, once the descent test (16) is satisfied at iteration $k-1$, we have

$$F_{\hat{x}}^k \le F_{\hat{x}}^{k-1} - \kappa v_{k-1} \le F_{\hat{x}}^{k-1} - \theta\kappa v_{k-1} = \gamma_x^k = \gamma_{\hat{x}}^k.$$

□

The following lemma shows that an (approximate) optimal solution can be obtained whenever the algorithm terminates finitely or loops infinitely between Steps 2 and 5.

Lemma 3. *If either Algorithm 1 terminates at the kth iteration at Step 4, or loops between Steps 2 and 5 infinitely, then*

(i) $\hat{x}^k$ *is an optimal solution to problem (1) for instances Ex and PI.*
(ii) $\hat{x}^k$ *is* ε*-optimal, that is,* $F(\hat{x}^k) \le F_* + \varepsilon$*, for instance IE.*
(iii) $\hat{x}^k$ *is* $\varepsilon_x^{k(l)}$*-optimal, that is,* $F(\hat{x}^k) \le F_* + \varepsilon_x^{k(l)}$*, for instances AE and PAE.*

Proof. Firstly, suppose that Algorithm 1 terminates at Step 4 with iteration k. Then from (23), we have $V_k = 0$. This together with (24) shows that

$$F_{\hat{x}}^k \le \inf\{F(x) : \ x \in \mathbb{R}^n\} = F_*. \tag{29}$$

Thus, from (7), we can conclude that: $F(\hat{x}^k) = F_{\hat{x}}^k \le F_*$ for instances Ex and PI; $F(\hat{x}^k) \le F_{\hat{x}}^k + \varepsilon \le F_* + \varepsilon$ for instance IE; $F(\hat{x}^k) \le F_{\hat{x}}^k + \varepsilon_x^{k(l)} \le F_* + \varepsilon_x^{k(l)}$ for instances AE and PAE.

Secondly, suppose that Algorithm 1 loops between Steps 2 and 5 infinitely. Then from Lemma 2 and the condition at Step 5, it follows that (28) holds and $t_k \uparrow \infty$. Thus, we obtain $V_k \to 0$. This along with (24) implies (29), and therefore the claims hold by repeating the corresponding lines in first case. □

From the above lemma, in what follows, we may assume that Algorithm 1 neither terminates finitely nor loops infinitely between Steps 2 and 5. In addition, as in Reference [22], we assume that the model subgradients $p_f^k \in \partial \check{f}_k(y^{k+1})$ in (17) satisfy that $\{p_f^k\}$ is bounded if $\{y^k\}$ is bounded.

Algorithm 1 must take only one of the following two cases:

(i) the algorithm generates finitely many descent steps;
(ii) the algorithm generates infinitely many descent steps.

We first consider case (i), in which two subcases may occur: $t_\infty := \lim_k t_k = \infty$ and $t_\infty < \infty$. The first subcase of $t_\infty = \infty$ is analyzed in the following lemma.

Lemma 4. *Suppose that Algorithm 1 generates finitely many descent steps, that is, there exists an index* $\bar{k}$ *such that only null steps occur for all* $k \ge \bar{k}$*, and that* $t_\infty = \infty$*. Denote* $\mathcal{K} := \{k \ge \bar{k} : t_{k+1} > t_k\}$*, then* $V_k \to 0$ *as* $k \in \mathcal{K}$*,* $k \to \infty$*.*

Proof. For the last time t_k increases before Step 5 for $k \in \mathcal{K}$, one has

$$V_k < \left(\frac{2\varepsilon_x^{\bar{k}}}{t_k}\right)^{1/2} \left(1 + \left\|\hat{x}^{\bar{k}}\right\|\right),$$

which along with $t_k \to \infty$ shows the lemma. □

The following lemma analyzes the second subcase of $t_\infty < \infty$.

Lemma 5. *Suppose that there exists $\bar{k}$ such that $\hat{x}^k = \hat{x}^{\bar{k}}$ and $t_{\min} \le t_{k+1} \le t_k$ for all $k \ge \bar{k}$. If the descent criterion (16) fails for all $k \ge \bar{k}$, then $V_k \to 0$.*

Proof. In view of the facts that $t_{\min} \le t_{k+1} \le t_k$ and $\hat{x}^k = \hat{x}^{\bar{k}}$ for all $k \ge \bar{k}$, we know that only null steps occur and t_k does not increase at Step 5. By Taylor's expansion, Cauchy-Schwarz inequality, and the properties of subproblems (11) and (13), we can conclude that $v_k \to 0$, so the conclusion holds from (27). For more details, one can refer to [11], Lemma 3.2. □

By combining Lemmas 4 and 5, we have the following lemma.

Lemma 6. *Suppose that there exists $\bar{k}$ such that only null steps occur for all $k \ge \bar{k}$. Let $\mathcal{K} := \{k \ge \bar{k} : t_{k+1} > t_k\}$ if $t_k \to \infty$; $\mathcal{K} := \{k : k \ge \bar{k}\}$ otherwise. Then $V_k \xrightarrow{\mathcal{K}} 0$.*

Now, we can present the main convergence result for the case where the algorithm generates finitely many descent steps.

Theorem 1. *Suppose that Algorithm 1 generates finitely many descent steps, and that $\hat{x}^{\bar{k}}$ is the last stability center. Then, $\hat{x}^{\bar{k}}$ is an optimal solution to problem (1) for instances Ex and PI; an ε-optimal solution for IE; and an $\varepsilon_x^{\bar{k}(l)}$-optimal solution for AE and PAE.*

Proof. Under the stated assumption, we know that $\hat{x}^k \equiv \hat{x}^{\bar{k}}$ and $f_{\hat{x}}^k \equiv f_{\hat{x}}^{\bar{k}}$ for all $k \ge \bar{k}$. This together with (24) and Lemma 6 shows that

$$F_{\hat{x}}^{\bar{k}} \le \inf\{F(x) : \ x \in \mathbb{R}^n\} = F_*.$$

Hence, similar to the proof of Lemma 3, we obtain the results of the theorem. □

Next, we consider the second case where the algorithm generates infinitely many descent steps.

Lemma 7. *Suppose that Algorithm 1 generates infinitely many descent steps, and that $F_{\hat{x}}^{\infty} := \lim_k F_{\hat{x}}^k > -\infty$. Let $K := \{k \ : F_{\hat{x}}^{k+1} < F_{\hat{x}}^k\}$. Then $v_k \xrightarrow{K} 0$ and $\underline{\lim}_{k\in K} V_k = 0$. Moreover, if $\{\hat{x}^k\}$ is bounded, then $V_k \xrightarrow{K} 0$.*

Proof. From the descent test condition (16), we may first prove that $v_k \xrightarrow{K} 0$, and therefore $\epsilon_k, t_k\|p_k\|^2 \xrightarrow{K} 0$ from (26) and $\|p_k\| \xrightarrow{K} 0$ from the fact that $t_k \ge t_{\min}$. It can be further proved that $\epsilon_k, \|p_k\| \xrightarrow{K} 0$, so it follows $\underline{\lim}_{k\in K} V_k = 0$ from the definition of V_k. Moreover, under the condition that $\{\hat{x}^k\}$ is bounded, we have $V_k \xrightarrow{K} 0$ by (v) of Lemma 1. For more details, one can refer to [11], Lemma 3.4. □

Finally, we present the convergence results for the second case.

Theorem 2. *Suppose that Algorithm 1 generates infinitely many descent steps, $F_{\hat{x}}^{\infty} > -\infty$, and that the index set K is defined in Lemma 7. Then*

(i) $F_* \le \underline{\lim}_{k\in K} F(\hat{x}^{k+1}) \le \overline{\lim}_{k\in K} F(\hat{x}^{k+1}) \le F_{\hat{x}}^{\infty} + \varepsilon$ *for instance IE in Table 1;*
(ii) $F_* \le \underline{\lim}_{k\in K} F(\hat{x}^{k+1}) \le \overline{\lim}_{k\in K} F(\hat{x}^{k+1}) \le F_{\hat{x}}^{\infty}$ *for the remaining instances in Table 1;*
(iii) $\underline{\lim}_{k\in K} V_k = 0$ *and* $F_{\hat{x}}^k \downarrow F_{\hat{x}}^{\infty} \le F_*$.

Proof. It is obvious that

$$F_* \le \underline{\lim}_{k\in K} F(\hat{x}^{k+1}) \le \overline{\lim}_{k\in K} F(\hat{x}^{k+1}). \tag{30}$$

(i) For instance IE, it follows that $\varepsilon_{\hat{x}}^{k+1} = \varepsilon$ and $F_{\hat{x}}^{k+1} \le \gamma_{\hat{x}}^{k+1} = +\infty$ for all $k \in K$. Then from (7), we have $F(\hat{x}^{k+1}) \le F_{\hat{x}}^{k+1} + \varepsilon, \forall k \in K$, which implies

$$\overline{\lim_{k\in K}} F(\hat{x}^{k+1}) \le \lim_{k\in K} F_{\hat{x}}^{k+1} + \varepsilon = F_{\hat{x}}^{\infty} + \varepsilon.$$

This along with (30) shows part (i).

(ii) Next, the other four instances in Table 1 are considered separately.

For instance Ex, we have $\varepsilon_{\hat{x}}^{k+1} = 0$, $F_{\hat{x}}^{k+1} \le \gamma_{\hat{x}}^{k+1} = +\infty$ and $F(\hat{x}^{k+1}) = F_{\hat{x}}^{k+1}$. This implies

$$\overline{\lim_{k\in K}} F(\hat{x}^{k+1}) = \lim_{k\in K} F_{\hat{x}}^{k+1} = F_{\hat{x}}^{\infty}.$$

For instance PI, we have $\varepsilon_{\hat{x}}^{k+1} = 0$ and $\gamma_{\hat{x}}^{k+1} = F_{\hat{x}}^{k} - \theta\kappa v_k$ for all $k \in K$. Thus, we obtain

$$F_{\hat{x}}^{k+1} \le F_{\hat{x}}^{k} - \kappa v_k \le F_{\hat{x}}^{k} - \theta\kappa v_k = \gamma_{\hat{x}}^{k+1}.$$

This implies $F(\hat{x}^{k+1}) = F_{\hat{x}}^{k+1}$, and therefore

$$\overline{\lim_{k\in K}} F(\hat{x}^{k+1}) = \lim_{k\in K} F_{\hat{x}}^{k+1} = F_{\hat{x}}^{\infty}.$$

For instance AE, we have $\varepsilon_{\hat{x}}^{k+1} = \kappa_\epsilon v_k$ and $F_{\hat{x}}^{k+1} \le \gamma_{\hat{x}}^{k+1} = +\infty$ for all $k \in K$, which implies

$$F(\hat{x}^{k+1}) \le F_{\hat{x}}^{k+1} + \varepsilon_{\hat{x}}^{k+1} \le F_{\hat{x}}^{k+1} + \kappa_\epsilon v_k.$$

This along with Lemma 7 ($v_k \xrightarrow{K} 0$) shows that

$$\overline{\lim_{k\in K}} F(\hat{x}^{k+1}) \le \lim_{k\in K} F_{\hat{x}}^{k+1} = F_{\hat{x}}^{\infty}.$$

For instance PAE, we have $\varepsilon_{\hat{x}}^{k+1} = \kappa_\epsilon v_k$ and $\gamma_{\hat{x}}^{k+1} = F_{\hat{x}}^{k} - \theta\kappa v_k$ for all $k \in K$. Then, it follows that

$$F_{\hat{x}}^{k+1} \le F_{\hat{x}}^{k} - \kappa v_k \le F_{\hat{x}}^{k} - \theta\kappa v_k = \gamma_{\hat{x}}^{k+1},$$

which implies

$$F(\hat{x}^{k+1}) \le F_{\hat{x}}^{k+1} + \kappa_\epsilon v_k.$$

Again from Lemma 7, we obtain

$$\overline{\lim_{k\in K}} F(\hat{x}^{k+1}) \le \lim_{k\in K} F_{\hat{x}}^{k+1} = F_{\hat{x}}^{\infty}.$$

Summarizing the above analysis and noticing (30), we complete the proof of part (ii).

(iii) From Lemma 7, we know that $\underline{\lim}_{k\in K} V_k = 0$. This together with (24) shows part (iii). □

5. Numerical Experiments

In this section, we aim to test the numerical efficiency of the proposed algorithm. In the fields of production and transportation, finance and insurance, power industry, and telecommunications, decision makers usually need to solve problems with uncertain information. As an effective tool to solve such problems, stochastic programming (SP) has attracted more and more attention and research on its practical instances and theories; see, for example, References [41,42]. We consider a class of two-stage SP problems with fixed recourse, whose discretization of uncertainty into N scenarios has the form (see e.g., References [23,43])

$$\begin{array}{ll} \min & f(x) := \langle c, x\rangle + \sum_{i=1}^{N} p_i V_i(x) \\ \text{s.t.} & x \in X := \{x \in \mathbb{R}_+^{n_1} : \ Ax = b\}, \end{array} \tag{31}$$

where x is the first-stage decision variable, $c \in \mathbb{R}^{n_1}$, $A \in \mathbb{R}^{m_1 \times n_1}$, and $b \in \mathbb{R}^{m_1}$. In addition, the recourse function is

$$V_i(x) := \min_{\pi \in R_+^{n_2}} \{\langle q, \pi \rangle : W\pi = h_i - T_i x\},$$

where corresponding to the ith scenario (h_i, T_i), with probability $p_i > 0$ for $h_i \in \mathbb{R}^{m_2}$ and $T_i \in \mathbb{R}^{m_2 \times n_1}$. Here π is the second-stage decision variable.

Clearly, by introducing the indicator function δ_X, problem (31) can be written as the form of (5), and then becomes the form of (1) by setting $h(x) = \delta_X(x)$.

The above recourse function can be written as its dual form:

$$V_i(x) = \max_{y \in R^{m_2}} \langle h_i - T_i x, y \rangle \quad \text{s.t.} \;\; W^T y \leq q,$$

where $q \in \mathbb{R}^{n_2}$ and $W \in \mathbb{R}^{m_2 \times n_2}$. By solving these linear programming problems to return solutions with precision up to a given tolerance, one can establish an inexact oracle in the form (6), see Reference [23] for more detailed description.

The instances of SP problems are downloaded from the link: http://pwp.gatech.edu/guanghui-lan/computer-codes/.

Four instances are tested, namely, SSN(50), SSN(100), 20-term(50), 20-term(100), where the integers in the brackets mean the number of scenarios N. Here, the SSN instances come from the telecommunications and have been studied by Sen, Doverspike, and Cosares [44]. And the 20-term instances come from the motor freight carrier's problem and have been studied by Mak, Morton, and Wood [45]. The dimensions of these instances are listed in Table 2.

Table 2. Dimensions of the SP instances.

Name	n_1	m_1	n_2	m_2
SSN	89	1	706	175
20-term	63	3	764	124

The parameters are selected as: $\kappa = 0.04$, $t_{min} = 0.1$ and $t_1 = 1.1$. The maximum bundle size is set to be 35. All the tests were performed in MATLAB (R2014a) on a PC with Intel(R) Core(TM) i7-4790 CPU 3.60GHz, 4GB RAM. The quadratic programming and linear programming subproblems were solved by the MOSEK 8 toolbox for MATLAB; see http://www.mosek.com.

We first compare our algorithm (denoted by GALBM) with the accelerated prox-level method (APL) in Reference [43], where the tolerances of the linear programming solver of MOSEK are set by default. The results are listed in Table 3, in which the number of iterations (NI), the consumed CPU time in seconds (Time), and the returned minimum values (f_*) are compared. Note that we use the MATLAB commands `tic` and `toc` to measure the consumed CPU time. For each instance, we run 10 times and report the average CPU time. From Table 3, we see that, when similar solution quality is achieved, GALBM can significantly outperform than APL in terms of the number of iterations and CPU time.

Table 3. The comparisons between GALBM and accelerated prox-level (APL) for the stochastic programming (SP) instances.

Name	Algorithm	NI	Time	f_*
SSN(50)	GALBM	105	28.12	4.838238
	APL	147	72.40	4.838278
SSN(100)	GALBM	95	49.58	7.352609
	APL	155	156.53	7.352618
20-term(50)	GALBM	132	47.62	2.549453×10^5
	APL	156	106.51	2.549453×10^5
20-term(100)	GALBM	173	128.23	2.532875×10^5
	APL	261	364.93	2.532876×10^5

In what follows, we are interested in evaluating the impact of inexact oracles for GALBM. In more detail, we carry out two groups of tests. The first group adopts fixed tolerances, that is, $\varepsilon_x^{k+1} \equiv \varepsilon$, and the corresponding results are reported in Tables 4–7. Whereas the second group adopts dynamic tolerances with a safeguard parameter $\mu > 0$, that is, $\varepsilon_x^{k+1} = \min\{\mu, \kappa_\epsilon v_k\}$ with $\kappa_\epsilon = 0.7$, and the corresponding results are reported in Tables 8–11. The symbol "-" in the following tables means that the number of iterations for the corresponding instance is greater than 500.

Table 4. Numerical results for SSN(50) with fixed tolerances.

No.	ε_x	NI	Time	f_*
1	10^{-4}	206	55.21	4.838157
2	10^{-5}	202	50.37	4.838163
3	10^{-6}	190	49.64	4.838247
4	10^{-7}	132	34.35	4.838156
5	10^{-8}	105	27.81	4.838238
6	10^{-9}	97	24.49	4.838188
7	10^{-10}	99	25.62	4.838136
8	10^{-11}	84	22.98	4.838191
9	10^{-12}	97	27.47	4.838128
10	10^{-13}	84	25.53	4.838231

Table 5. Numerical results for SSN(100) with fixed tolerances.

No.	ε_x	NI	Time	f_*
1	10^{-4}	-	-	-
2	10^{-5}	224	109.21	7.352932
3	10^{-6}	194	94.80	7.352854
4	10^{-7}	127	62.17	7.352937
5	10^{-8}	95	46.93	7.352758
6	10^{-9}	87	43.03	7.352750
7	10^{-10}	79	41.55	7.353074
8	10^{-11}	83	43.76	7.352939
9	10^{-12}	81	44.08	7.352748
10	10^{-13}	81	46.77	7.352734

Table 6. Numerical results for 20-term (50) with fixed tolerances.

No.	ε_x	NI	Time	f_*
1	10^{-2}	250	80.86	2.549490×10^5
3	10^{-3}	144	49.12	2.549466×10^5
3	10^{-4}	165	57.49	2.549463×10^5
4	10^{-5}	164	54.12	2.549460×10^5
5	10^{-6}	211	72.42	2.549461×10^5
6	10^{-7}	178	62.19	2.549459×10^5
7	10^{-8}	132	44.56	2.549457×10^5
8	10^{-9}	175	61.53	2.549461×10^5
9	10^{-10}	132	49.53	2.549457×10^5
10	10^{-11}	258	99.69	2.549457×10^5
11	10^{-12}	175	67.70	2.549461×10^5
12	10^{-13}	183	71.52	2.549461×10^5

Table 7. Numerical results for 20-term (100) with fixed tolerances.

No.	ε_x	NI	Time	f_*
1	10^{-2}	227	141.74	2.532914×10^5
2	10^{-3}	-	-	-
3	10^{-4}	140	96.02	2.532879×10^5
4	10^{-5}	179	117.71	2.532877×10^5
5	10^{-6}	152	99.29	2.532879×10^5
6	10^{-7}	139	95.07	2.532876×10^5
7	10^{-8}	173	128.44	2.532876×10^5
8	10^{-9}	143	103.51	2.532878×10^5
9	10^{-10}	-	-	-
10	10^{-11}	159	110.21	2.532877×10^5
11	10^{-12}	150	112.37	2.532878×10^5
12	10^{-13}	132	99.46	2.532878×10^5

Table 8. Numerical results for SSN (50) with dynamic tolerances.

No.	μ	NI	Time	f_*
1	10^{-3}	-	-	-
2	10^{-4}	201	50.54	4.838163
3	10^{-5}	202	49.07	4.838163
4	10^{-6}	190	49.21	4.838247
5	10^{-7}	132	32.26	4.838156
6	10^{-8}	105	27.10	4.838238

Table 9. Numerical results for SSN (100) with dynamic tolerances.

No.	μ	NI	Time	f_*
1	10^{-3}	-	-	-
2	10^{-4}	-	-	-
3	10^{-5}	284	141.85	7.353010
4	10^{-6}	194	97.01	7.352854
5	10^{-7}	127	63.83	7.352937
6	10^{-8}	95	47.97	7.352758

Table 10. Numerical results for 20-term (50) with dynamic tolerances.

No.	μ	NI	Time	f_*
1	10^{-2}	199	65.09	2.549485×10^5
2	10^{-3}	140	44.73	2.549462×10^5
3	10^{-4}	165	54.08	2.549463×10^5
4	10^{-5}	164	54.18	2.549460×10^5
5	10^{-6}	211	70.67	2.549461×10^5
6	10^{-7}	178	61.90	2.549459×10^5
7	10^{-8}	132	46.94	2.549457×10^5

Table 11. Numerical results for 20-term (100) with dynamic tolerances.

No.	μ	NI	Time	f_*
1	10^{-2}	191	119.40	2.532901×10^5
2	10^{-3}	143	92.32	2.532881×10^5
3	10^{-4}	140	91.68	2.532878×10^5
4	10^{-5}	179	121.12	2.532877×10^5
5	10^{-6}	146	104.27	2.532878×10^5
6	10^{-7}	170	120.29	2.532879×10^5
7	10^{-8}	173	126.57	2.532876×10^5

6. Conclusions

In this paper, we have proposed a generalized alternating linearization bundle method for solving structured convex optimization with inexact first-order oracles. Our method can handle various inexact data by making use of the so-called on-demand accuracy oracles. At each iteration, two interrelated subproblems are solved alternately, aiming to reduce the computational cost. Global convergence of the algorithm is established under different types of inexactness. Numerical results show that the proposed algorithm is promising.

Author Contributions: C.T. mainly contributed to the algorithm design and convergence analysis; Y.L. and X.D. mainly contributed to the convergence analysis and numerical results; and B.H. mainly contributed to the numerical results. All authors have read and agree to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation (11761013) and Guangxi Natural Science Foundation (2018GXNSFFA281007) of China.

Acknowledgments: The authors would like to thank for the support funds.

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

References

1. Tsaig, Y.; Donoho, D.L. Compressed sensing. *IEEE Trans. Inform. Theory* **2006**, *52*, 1289–1306.
2. Jung, M.; Kang, M. Efficient nonsmooth nonconvex optimization for image restoration and segmentation. *J. Sci. Comput.* **2015**, *62*, 336–370. [CrossRef]
3. Sar, S.; Nowozin, S.; Wright, S.J. *Optimization for Machine Learning*; Massachusetts Institute of Technology Press: Cambridge, MA, USA, 2012.
4. Clarke, F.H.; Ledyaev, Y.S.; Stern, R.J.; Wolenski, P.R. *Nonsmooth Analysis and Control Theory*; Springer: New York, NY, USA, 1998.
5. Yang, L.F.; Luo, J.Y.; Jian, J.B.; Zhang, Z.R.; Dong, Z.Y. A distributed dual consensus ADMM based on partition for DC-DOPF with carbon emission trading. *IEEE Trans. Ind. Inform.* **2020**, *16*, 1858–1872. [CrossRef]
6. Yang, L.F.; Zhang, C.; Jian, J.B.; Meng, K.; Xu, Y.; Dong, Z.Y. A novel projected two-binary-variable formulation for unit commitment in power systems. *Appl. Energ.* **2017**, *187*, 732–745. [CrossRef]
7. Yang, L.F.; Jian, J.B.; Zhu, Y.N.; Dong, Z.Y. Tight relaxation method for unit commitment problem using reformulation and lift-and-project. *IEEE Trans. Power. Syst.* **2015**, *30*, 13–23. [CrossRef]

8. Yang, L.F.; Jian, J.B.; Xu, Y.; Dong, Z.Y.; Ma, G.D. Multiple perspective-cuts outer approximation method for risk-averse operational planning of regional energy service providers. *IEEE Trans. Ind. Inform.* **2017**, *13*, 2606–2619. [CrossRef]
9. Teo, C.H.; Vishwanathan, S.V.N.; Smola, A.J.; Le, Q.V. Bundle methods for regularized risk minimization. *Mach. Learn. Res.* **2010**, *11*, 311–365.
10. Bottou, L.; Curtis, F.E.; Nocedal, J. Optimization Methods for Machine Learning. *SIAM Rev.* **2018**, *60*, 223–311. [CrossRef]
11. Kiwiel, K.C. A proximal-projection bundle method for Lagrangian relaxation, including semidefinite programming. *SIAM J. Optim.* **2006**, *17*, 1015–1034. [CrossRef]
12. Tang, C.M.; Jian, J.B.; Li, G.Y. A proximal-projection partial bundle method for convex constrained minimax problems. *J. Ind. Manag. Optim.* **2019**, *15*, 757–774. [CrossRef]
13. Paul, T. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Cont. Optim.* **1991**, *29*, 119–138.
14. Mahey, P.; Tao, P.D. Partial regularization of the sum of two maximal monotone operators. *Math. Model. Numer. Anal.* **1993**, *27*, 375–392. [CrossRef]
15. Eckstein, J. Some saddle-function splitting methods for convex programming. *Optim. Meth. Soft.* **1994**, *4*, 75–83. [CrossRef]
16. Fukushima, M. Application of the alternating direction method of multipliers to separable convex programming problems. *Comput. Optim. Appl.* **1992**, *1*, 93–111. [CrossRef]
17. He, B.S.; Tao, M.; Yuan, X.M. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM J. Optim.* **2012**, *22*, 313–340. [CrossRef]
18. He, B.S.; Tao, M.; Yuan, X.M. Convergence rate analysis for the alternating direction method of multipliers with a substitution procedure for separable convex. *Math. Oper. Res.* **2017**, *42*, 662–691. [CrossRef]
19. Chao, M.; Cheng, C.; Zhang, H. A linearized alternating direction method of multipliers with substitution procedure. *Asia-Pac. Opera. Res.* **2015**, *32*, 1550011. [CrossRef]
20. Goldfarb, D.; Ma, S.; Scheinberg, K. Fast alternating linearization methods for minimizing the sum of two convex functions. *Math. Program* **2013**, *141*, 349–382. [CrossRef]
21. Kiwiel, K.C.; Rosa, C.H.; Ruszczyński, A. Proximal decomposition via alternating linearization. *SIAM J. Optim.* **1999**, *9*, 668–689. [CrossRef]
22. Kiwiel, K.C. An alternating linearization bundle method for convex optimization and nonlinear multicommodity flow problems. *Math. Program* **2011**, *130*, 59–84. [CrossRef]
23. DeOliveira, W.; Sagastizábal, C. Level bundle methods for oracles with on-demand accuracy. *Optim. Method Softw.* **2014**, *29*, 1180–1209. [CrossRef]
24. Kiwiel, K.C. *Bundle Methods for Convex Minimization with Partially Inexact Oracles*; Technical Report; Systems Research Institute, Polish Academy of Sciences: Warsaw, Poland, 2009.
25. De Oliveira, W.; Sagastizábal, C.; Lemaréchal, C. Convex proximal bundle methods in depth: A unified analysis for inexact oracles. *Math. Program.* **2014**, *148*, 241–277. [CrossRef]
26. Wolfe, P. A method of conjugate subgradients for minimizing nondifferentiable functions. *Math. Program.* **1975**, *3*, 145–173.
27. Mäkelä, M. Survey of bundle methods for nonsmooth optimization. *Optim. Meth. Soft.* **2002**, *17*, 1–29. [CrossRef]
28. Bonnans, J.F.; Gilbert, J.C.; Lemaréchal, C.; Sagastizábal, C. *Numerical Optimization: Theoretical and Practical Aspects*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2006.
29. Lemaréchal, C. An extension of davidon methods to nondifferentiable problems. *Math. Program.* **1975**, *3*, 95–109.
30. Kiwiel, K.C. Approximations in proximal bundle methods and decomposition of convex programs. *J. Optim. Theory. Appl.* **1995**, *84*, 529–548. [CrossRef]
31. Hintermuller, M. A proximal bundle method based on approximate subgradients. *Comput. Optim. Appl.* **2001**, *20*, 245–266. [CrossRef]
32. Kiwiel, K.C. A proximal bundle method with approximate subgradient linearizations. *SIAM J. Optim.* **2006**, *16*, 1007–1023. [CrossRef]
33. Kiwiel, K.C. An algorithm for nonsmooth convex minimization with errors. *Math. Comput.* **1985**, *45*, 173–180. [CrossRef]

34. Hiriart-Urruty, J.B.; Lemaréchal, C. *Convex Analysis and Minimization Algorithms. Number 305-306 in Grundlehren der mathematischen Wissenschaften*; Springer: Berlin, Germany, 1993.
35. Rockafellar, R.T. *Convex Analysis*; Princeton University Press: Princeton, NJ, USA, 2015.
36. Malick, J.; De Oliveira, W.; Zaourar-Michel, S. Uncontrolled inexact information within bundle methods. *Eur. J. Oper. Res.* **2017**, *5*, 5–29. [CrossRef]
37. De Oliveira, W.; Sagastizábal, C.; Scheimberg, S. Inexact bundle methods for two-stage stochastic programming. *SIAM J. Optim.* **2011**, *21*, 517–544. [CrossRef]
38. Zakeri, G.; Philpott, A.B.; Ryan, D.M. Inexact cuts in benders decomposition. *SIAM J. Optim.* **2000**, *10*, 643–657. [CrossRef]
39. Fábixaxn, C.I. Bundle-type methods for inexact data. *Cent. Eur. J. Oper. Res.* **2000**, *8*, 35–55.
40. Kiwiel, K.C. *Methods of Descent for Nondifferentiable Optimization*; Lecture Notes in Mathematics; Springer: Berlin, Germany, 1985.
41. Wallace, S.W.; Ziemba, W.T. *Applications of Stochastic Programming*; MPS-SIAM Ser. Optim.; SIAM: Philadelphia, PA, USA, 2005.
42. Shapiro, A.; Dentcheva, D.; Ruszczyński, A. *Lectures on Stochastic Programming: Modeling and Theory*; SIAM: Philadelphia, PA, USA, 2014.
43. Lan, G. Bundle-level type methods uniformly optimal for smooth and nonsmooth convex optimization. *Math. Program* **2015**, *149*, 1–45. [CrossRef]
44. Sen, S.; Doverspike, R.D.; Cosares, S. Network planning with random demand. *Telecommun. Syst.* **1994**, *3*, 11–30. [CrossRef]
45. Mak, W.; Morton, D.P.; Wood, R.K. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Oper. Res. Lett.* **1999**, *24*, 47–56. [CrossRef]