Mathematical Programming manuscript No.
(will be inserted by the editor)

Napsu Haarala · Kaisa Miettinen · Marko M. Mäkelä

Globally convergent limited memory bundle method for large-scale nonsmooth optimization

Received: date / Accepted: date

Abstract Many practical optimization problems involve nonsmooth (that is, not necessarily differentiable) functions of thousands of variables. In the paper [Haarala, Miettinen, Mäkelä, Optimization Methods and Software, 19, (2004), pp. 673–692] we have described an efficient method for large-scale nonsmooth optimization. In this paper, we introduce a new variant of this method and prove its global convergence for locally Lipschitz continuous objective functions, which are not necessarily differentiable or convex. In addition, we give some encouraging results from numerical experiments.

Keywords Nondifferentiable programming · large-scale optimization · bundle methods · limited memory variable metric methods · global convergence

1 Introduction

In this paper, we provide a global convergence theory for a limited memory bundle algorithm [11] for solving large-scale nonsmooth (nondifferentiable)

N. Haarala
School of Computational & Applied Mathematics, University of the Witwatersrand, Private Bag 3, Wits 2050, Johannesburg, South Africa
E-mail: mshaarala@cam.wits.ac.za

K. Miettinen
Helsinki School of Economics, P.O. Box 1210, FI-00101 Helsinki, Finland
E-mail: kaisa.miettinen@hse.fi

M.M Mäkelä
Department of Mathematical Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland
E-mail: makela@mit.jyu.fi

unconstrained optimization problems. Thus, we consider the problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is supposed to be locally Lipschitz continuous and the number of variables n is supposed to be large.

Nonsmooth optimization problems of type (1) can be encountered in many fields of applications, for instance, in optimal control [5], engineering [26], mechanics [27], and economics [31]. Moreover, nonsmooth optimization techniques can be successfully applied to smooth problems but not vice versa (see, e.g., [17]) and, thus, we can say that nonsmooth optimization deals with a broader class of problems than smooth optimization.

Various versions of bundle methods (see, e.g., [12,15,22,24,33]) are regarded as the most effective and reliable globally convergent methods for nonsmooth optimization (see, e.g., [22]) at the moment. They assume that at every point $\boldsymbol{x} \in \mathbb{R}^n$ we can evaluate the value of the objective function $f(\boldsymbol{x})$ and an arbitrary subgradient $\boldsymbol{\xi} \in \mathbb{R}^n$ from the subdifferential [4]

$$\partial f(\boldsymbol{x}) = \text{conv}\{ \lim_{i \to \infty} \nabla f(\boldsymbol{x}_i) \mid \boldsymbol{x}_i \to \boldsymbol{x} \text{ and } \nabla f(\boldsymbol{x}_i) \text{ exists} \},$$

where "conv" denotes the convex hull of a set. The idea of bundle methods is to approximate the subdifferential by gathering subgradients from previous iterations into a bundle.

In some cases (see, e.g., convex problems in [1,9,30]) standard bundle methods may work well even for very large problems. However, often the computational demand of bundle methods expands already with relatively small problems (see, e.g., [11,14]). This is due to the fact that the complexity of the problem depends not only on the number of variables but also on the nature (that is, linearity, convexity, relaxation properties, etc.) of the problem.

In [11] we have introduced a limited memory bundle method for general, possibly nonconvex, nonsmooth large-scale optimization. The method is a hybrid of the variable metric bundle methods [20,34] and the limited memory variable metric methods (see, e.g., [3,8,19,29]), where the first ones have been developed for small- and medium-scale nonsmooth optimization and the latter ones, on the contrary, for smooth large-scale optimization. The new method exploits the ideas of the variable metric bundle methods, namely the utilization of null steps, simple aggregation of subgradients, and the subgradient locality measures, but the search direction is calculated using a limited memory approach. Therefore, the time-consuming quadratic direction finding problem appearing in standard bundle methods (see, e.g., [15,24,33]) need not to be solved and the number of stored subgradients is independent of the dimension of the problem. Furthermore, the method uses only few vectors to represent the variable metric updates and, thus, it avoids storing and manipulating large matrices as is the case in variable metric bundle methods [20,34].

In order to prove the global convergence of the limited memory bundle method, some modifications had to be made to the algorithm introduced

in [11]. In this paper, we first present a modified limited memory bundle algorithm for which we can prove the global convergence. Then, we propose a special line search procedure, which is modified from that given in [34] and used in [11]. In addition, we give a modified limited memory SR1 update, which guarantees the conditions required for the global convergence in the consecutive null steps. The limited memory BFGS update used is the same as in [11] and it has no effect on the global convergence of the method. For a more detailed description of the BFGS update and other properties of the limited memory bundle method we refer to [11].

The rest of this paper is organized as follows. We describe the modified limited memory bundle algorithm together with the line search procedure and the limited memory matrix updating in Section 2. In Section 3, we prove the global convergence of the method and, in Section 4, some results of numerical experiments are presented. The problems included in our experiments contain both academic test problems and practical applications arising in the field of nonsmooth large-scale optimization. The numerical results demonstrate the usability and the reliability of the limited memory bundle method with both convex and nonconvex large-scale nonsmooth minimization problems. Finally, in Section 5, we conclude and give some ideas of further development.

2 Limited Memory Bundle Method

In this section, we first describe the modified limited memory bundle algorithm. After that, we present the line search procedure, which is used to determine step sizes in the limited memory bundle method. Finally, at the end of this section, we consider the limited memory matrix updating and give an efficient algorithm for direction finding using limited memory SR1 update.

Limited memory bundle method. We start this section with a simple flowchart of the limited memory bundle method (in Figure 1) to point out the basic ideas of the algorithm. The limited memory bundle method is characterized by the usage of null steps together with the aggregation of subgradients. Moreover, the search direction is calculated by using the limited memory variable metric updates (the limited memory BFGS update after a serious step and the limited memory SR1 update, otherwise). Using null steps gives sufficient information about the nonsmooth objective function in the cases the descent condition (3) is not satisfied. On the other hand, a simple aggregation of subgradients guarantees the convergence of the aggregate subgradients to zero and makes it possible to evaluate a termination criterion.

Next, we give a more detailed description of the method. The limited memory bundle algorithm to be presented generates a sequence of basic points $(\boldsymbol{x}_k) \subset \mathbb{R}^n$ that, in the convex case, converges to a global minimum of an objective function $f : \mathbb{R}^n \to \mathbb{R}$. In the nonconvex case, the algorithm is only guaranteed to find a stationary point of the objective function (that is, a point $\boldsymbol{x} \in \mathbb{R}^n$ satisfying $\mathbf{0} \in \partial f(\boldsymbol{x})$). In addition to the basic points, the algorithm generates a sequence of auxiliary points $(\boldsymbol{y}_k) \subset \mathbb{R}^n$. A new

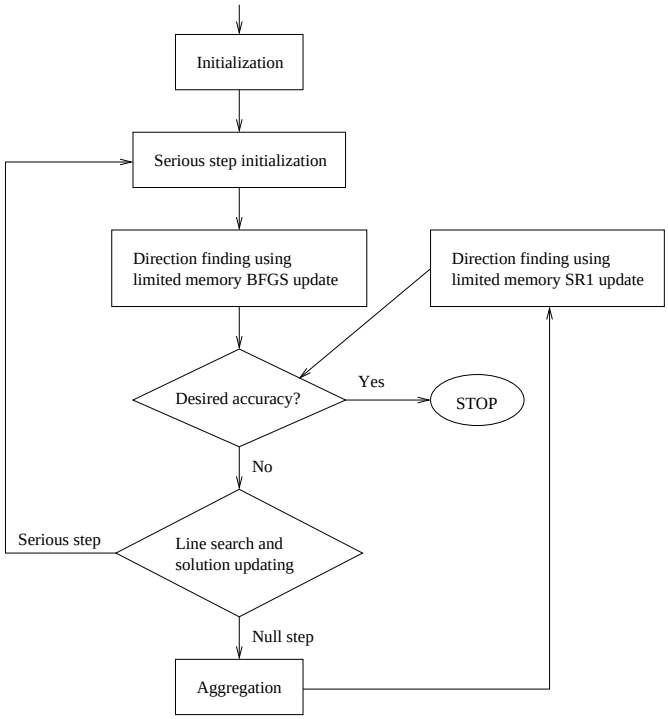


Fig. 1 Flowchart of the limited memory bundle method.

iteration point $\boldsymbol{x}_{k+1}$ and a new auxiliary point $\boldsymbol{y}_{k+1}$ are produced by using a special line search procedure such that

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k \qquad \text{and} \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k, \qquad \text{for } k \geq 1 \end{aligned} \tag{2}$$

with $\boldsymbol{y}_1 = \boldsymbol{x}_1$, where $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$ are step sizes, $t_{max} > 1$ is the upper bound for the step size, $\boldsymbol{d}_k = -D_k \tilde{\boldsymbol{\xi}}_k$ is a search direction vector, $\tilde{\boldsymbol{\xi}}_k$ is an aggregate subgradient, and D_k is the limited memory variable metric update that, in the smooth case, represents the approximation of the inverse of the Hessian matrix. Note that the matrix D_k is not formed explicitly but the search direction $\boldsymbol{d}_k$ is calculated using the limited memory approach to be described later.

A necessary condition for a *serious step* to be taken is to have

$$t_R^k = t_L^k > 0 \qquad \text{and} \qquad f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L^k t_R^k w_k, \tag{3}$$

where $\varepsilon_L^k \in (0, 1/2)$ is a line search parameter and $w_k > 0$ represents the desirable amount of descent of f at $\boldsymbol{x}_k$. If condition (3) is satisfied, we set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ and a serious step is taken.

On the other hand, a *null step* is taken if

$$t_R^k > t_L^k = 0 \qquad \text{and} \qquad -\beta_{k+1} + \boldsymbol{d}_k^T \boldsymbol{\xi}_{k+1} \geq -\varepsilon_R^k w_k, \tag{4}$$

where $\varepsilon_R^k \in (\varepsilon_L^k, 1/2)$ is a line search parameter, $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, and β_{k+1} is the subgradient locality measure [18,25] similar to bundle methods. In the case of a null step, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ but information about the objective function is increased because we store the auxiliary point $\boldsymbol{y}_{k+1}$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$.

Similarly to the original variable metric bundle methods [20,34], the aggregation procedure used with the limited memory bundle method utilizes only three subgradients and two locality measures. Let us denote by m the lowest index j satisfying $\boldsymbol{x}_j = \boldsymbol{x}_k$ (that is, m is the index of the iteration after the latest serious step) and suppose that we have the current subgradient $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$, the auxiliary subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, and the current aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$ (note that $\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$) available. Now, the new aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k+1}$ is defined as a convex combination

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k,$$

where the multipliers λ_i^k satisfying $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$ and $\sum_{i=1}^3 \lambda_i^k = 1$ can be determined by minimizing a simple quadratic function, which depends on these three subgradients and two locality measures (see Step 6 in Algorithm 1). This simple aggregation procedure gives us a possibility to retain the global convergence without solving the quite complicated quadratic direction finding problem appearing in standard bundle methods. Note that the aggregate values are computed only if the last step was a null step. Otherwise, we set $\tilde{\boldsymbol{\xi}}_{k+1} = \boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{x}_{k+1})$.

We now present the limited memory bundle method for nonsmooth large-scale unconstrained optimization.

Algorithm 1 *(Limited Memory Bundle Method).*

Data: Choose the final accuracy tolerance $\varepsilon > 0$, the positive initial line search parameters $\varepsilon_L^I \in (0, 1/2)$ and $\varepsilon_R^I \in (\varepsilon_L^I, 1/2)$, the distance measure parameter $\gamma \geq 0$ (with $\gamma = 0$ if f is convex), and the locality measure parameter $\omega \geq 1$. Select the lower and the upper bounds $t_{min} \in (0, 1)$ and $t_{max} > 1$ for serious steps. Select a control parameter $C > 0$ for the length of the direction vector and a correction parameter $\varrho \in (0, 1/2)$.

Step 0: (*Initialization.*) Choose a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$ and set an initial matrix $D_1 = I$. Set $\boldsymbol{y}_1 = \boldsymbol{x}_1$ and $\beta_1 = 0$. Compute $f_1 = f(\boldsymbol{x}_1)$ and $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$. Set the correction indicator $i_C = 0$ and the iteration counter $k = 1$.

Step 1: (*Serious step initialization.*) Set the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k$ and the aggregate subgradient locality measure $\tilde{\beta}_k = 0$. Set the correction indicator $i_{CN} = 0$ for consecutive null steps and an index for the serious step $m = k$.

Step 2: (*Direction finding.*) Compute

$$\boldsymbol{d}_k = -D_k \tilde{\boldsymbol{\xi}}_k \tag{5}$$

by using a limited memory BFGS update if $m = k$ and by using a limited memory SR1 update, otherwise. Note that $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$.

Step 3: (*Correction.*) If $-\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k < \varrho \tilde{\boldsymbol{\xi}}_k^T \tilde{\boldsymbol{\xi}}_k$ or $i_{CN} = 1$, then set

$$\boldsymbol{d}_k = \boldsymbol{d}_k - \varrho \tilde{\boldsymbol{\xi}}_k, \tag{6}$$

(i.e., $D_k = D_k + \varrho I$) and $i_C = 1$. Otherwise, set $i_C = 0$. If $i_C = 1$ and $m < k$, then set $i_{CN} = 1$.

Step 4: (*Stopping criterion.*) Set

$$w_k = -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k + 2\tilde{\beta}_k \qquad \text{and} \tag{7}$$

$$q_k = \frac{1}{2} \tilde{\boldsymbol{\xi}}_k^T \tilde{\boldsymbol{\xi}}_k + \tilde{\beta}_k. \tag{8}$$

If $w_k < \varepsilon$ and $q_k < \varepsilon$, then stop with $\boldsymbol{x}_k$ as the final solution.

Step 5: (*Line search.*) Set the scaling parameter for the length of the direction vector and for line search

$$\theta_k = \min\{\, 1, C/\|\boldsymbol{d}_k\| \,\}.$$

Calculate the initial step size $t_I^k \in [t_{min}, t_{max})$. Determine the step sizes $t_R^k \in (0, t_I^k]$ and $t_L^k \in [0, t_R^k]$ by the line search Algorithm 2. Set the corresponding values

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \theta_k \boldsymbol{d}_k, \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \theta_k \boldsymbol{d}_k, \\ f_{k+1} &= f(\boldsymbol{x}_{k+1}), \qquad \text{and} \\ \boldsymbol{\xi}_{k+1} &\in \partial f(\boldsymbol{y}_{k+1}). \end{aligned}$$

Set $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ and $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k = t_R^k \theta_k \boldsymbol{d}_k$. If condition (3) is valid (i.e., we take a serious step), then set $\beta_{k+1} = 0$, $k = k + 1$, and go to Step 1. Otherwise (i.e., condition (4) is valid), calculate the locality measure

$$\beta_{k+1} = \max\{|f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + \boldsymbol{s}_k^T \boldsymbol{\xi}_{k+1}|,\, \gamma \|\boldsymbol{s}_k\|^\omega \,\}. \tag{9}$$

Step 6: (*Aggregation.*) Determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function

$$\begin{aligned} \varphi(\lambda_1, \lambda_2, \lambda_3) = \; & (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k)^T D_k (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k) \\ & + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k), \end{aligned} \tag{10}$$

where D_k is calculated by the same updating formula as in Step 2 and $D_k = D_k + \varrho I$ if $i_C = 1$. Set

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad \text{and} \tag{11}$$

$$\tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{12}$$

Set $k = k + 1$ and go to Step 2.

In order to guarantee the global convergence of the method, the boundedness of both the length of the direction vector (see Step 5 in Algorithm 1) and the matrices $B_i = D_i^{-1}$ (see Step 3 in Algorithm 1) are required (we say that a matrix is bounded if its eigenvalues lie in the compact interval that does not contain zero). The utilization of correction (6) is equivalent to adding a positive definite matrix ϱI to matrix D_k. Note that in Steps 2 and 6, the matrices D_k are not formed explicitly but the limited memory expressions (14) and (16) are used instead.

The initial step size $t_I^k \in [t_{min}, t_{max})$ (see Step 5 in Algorithm 1) is selected by using a bundle containing auxiliary points and the corresponding function values and subgradients. The procedure used is exactly the same as in the original variable metric bundle method for nonconvex objective functions [34]. Since the aggregation procedure (see Step 6 in Algorithm 1) uses only three subgradients, the minimum size of the bundle is two and a larger bundle (if it is employed) is used only for the selection of this initial step size.

Line search procedure. Next we present a line search algorithm, which is used to determine the step sizes t_L^k and t_R^k in the limited memory bundle method. The line search procedure used is rather similar to that given in [34]. However, in order to guarantee the global convergence of the method, we use scaled line search parameters ε_L^k, ε_R^k, ε_A^k, and ε_T^k instead of fixed ones. Furthermore, in order to avoid many consecutive null steps, we have added an additional interpolation step (at Step 3 in Algorithm 2). In other words, we look for more suitable step sizes t_L^k and t_R^k by using an extra interpolation loop if necessary. The role of this additional step is that if we have already taken a null step at the previous iteration (that is, $i_{null} = k - m \geq 1$), we rather try to find a step size suitable for a serious step (that is, to make condition (3) valid) even if condition (4) required for a null step was satisfied. This additional interpolation step has no influence on the convergence properties but it has a significant effect on the efficiency of the method. The choice of the interpolation procedure (see Step 5 in Algorithm 2) has no effect on the convergence properties, either. Similarly to the original variable metric bundle method [34] we combine quadratic interpolation with the bisection procedure.

Algorithm 2 *(Line Search).*

Data: Suppose that we have the current iteration point $\boldsymbol{x}_k$, the current search direction $\boldsymbol{d}_k$, the current scaling parameter $\theta_k \in (0, 1]$, and the positive initial line search parameters $\varepsilon_L^I \in (0, 1/2)$, $\varepsilon_R^I \in (\varepsilon_L^I, 1/2)$, $\varepsilon_A^I \in (0, \varepsilon_R^I - \varepsilon_L^I)$, and $\varepsilon_T^I \in (\varepsilon_L^I, \varepsilon_R^I - \varepsilon_A^I)$. Suppose also that we have the initial step size t_I^k, an auxiliary lower bound for serious steps $t_{min} \in (0, 1)$, the distance measure parameter $\gamma \geq 0$ (with $\gamma = 0$ if f is convex), the locality measure parameter $\omega \geq 1$, the desirable amount of descent w_k, and the maximum number of additional interpolations i_{max} available. In addition, suppose that we have the number of consecutive null steps $i_{null} \geq 0$.

Step 0: (*Initialization.*) Set $t_A = 0$, $t = t_U = t_I^k$, and $i_I = 0$. Calculate the scaled line search parameters

$$\varepsilon_L^k = \theta_k \varepsilon_L^I, \quad \varepsilon_R^k = \theta_k \varepsilon_R^I, \quad \varepsilon_A^k = \theta_k \varepsilon_A^I, \quad \text{and} \quad \varepsilon_T^k = \theta_k \varepsilon_T^I$$

and the interpolation parameter

$$\kappa = 1 - \frac{1}{2(1-\varepsilon_T^k)}.$$

Step 1: (*New values.*) Compute $f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k)$, $\boldsymbol{\xi} \in \partial f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k)$, and

$$\beta = \max\{\, |f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) + t\theta_k \boldsymbol{d}_k^T \boldsymbol{\xi}|,\, \gamma\, (t\theta_k \|\boldsymbol{d}_k\|)^\omega \,\}.$$

If $f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) - \varepsilon_T^k t w_k$, then set $t_A = t$. Otherwise, set $t_U = t$.

Step 2: (*Serious step.*) If

$$f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) - \varepsilon_L^k t w_k,$$

and either

$$t \geq t_{min} \qquad \text{or} \qquad \beta > \varepsilon_A^k w_k,$$

then set $t_R^k = t_L^k = t$ and stop.

Step 3: (*Test for additional interpolation.*) If $f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) > f(\boldsymbol{x}_k)$, $i_{null} > 0$, and $i_I < i_{max}$, then set $i_I = i_I + 1$ and go to Step 5.

Step 4: (*Null step.*) If

$$-\beta + \theta_k \boldsymbol{d}_k^T \boldsymbol{\xi} \geq -\varepsilon_R^k w_k,$$

then set $t_R^k = t$, $t_L^k = 0$ and stop.

Step 5: (*Interpolation.*) If $t_A = 0$, then set

$$t = \max\left\{ \kappa t_U, \frac{-\frac{1}{2} t_U^2 w_k}{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) - t_U w_k} \right\}.$$

Otherwise, set $t = \frac{1}{2}(t_A + t_U)$. Go to Step 1.

It can be proved under some semi-smoothness assumptions [2] that Algorithm 2 terminates in a finite number of iterations (the proof is similar to that given in [34]). In addition, on the output of Algorithm 2 (see Steps 2 and 4), the step sizes t_L^k and t_R^k satisfy the serious descent criterion

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq -\varepsilon_L^k t_L^k w_k \tag{13}$$

and, in case of $t_L^k = 0$ (null step), also condition (4). Note that, after each iteration the interval containing t (that is, $t \in [t_A + \kappa(t_U - t_A), t_U - \kappa(t_U - t_A)]$) is reduced, (see, e.g., [24]).

Matrix updating. Finally, we need to consider how to update the matrix D_k and, at the same time, how to find the search direction $\boldsymbol{d}_k$. The basic idea in direction finding is the same as with the limited memory variable metric methods (see, e.g., [3]). However, due to the usage of null steps some modifications similar to the variable metric bundle methods [20,34] have to be made: If the previous step was a null step, the matrix D_k is formed by using the limited memory SR1 update (16), since this update formula gives a possibility to preserve the boundedness and some other properties of generated matrices that are required in the proof of global convergence. Otherwise, since these properties are not required after a serious step, the more efficient limited memory BFGS update formula (14) is employed.

The idea of the limited memory matrix updating is that instead of storing the matrices D_k, we use information of the last few iterations to implicitly define the variable metric update. Thus, at every iteration, we store a certain (small) number of correction pairs $(\boldsymbol{s}_i, \boldsymbol{u}_i)$, $(i < k)$, obtained in Step 5 of Algorithm 1.

Let us denote by $\hat{m}_c$ the user-specified maximum number of stored correction pairs $(3 \leq \hat{m}_c)$ and by $\hat{m}_k = \min\{\, k-1, \hat{m}_c \,\}$ the current number of stored correction pairs. Then, the $n \times \hat{m}_k$ dimensional correction matrices S_k and U_k are defined by

$$S_k = \begin{bmatrix} \boldsymbol{s}_{k-\hat{m}_k} & \dots & \boldsymbol{s}_{k-1} \end{bmatrix} \qquad \text{and}$$
$$U_k = \begin{bmatrix} \boldsymbol{u}_{k-\hat{m}_k} & \dots & \boldsymbol{u}_{k-1} \end{bmatrix}.$$

When the storage space available is used up, the oldest correction vectors are deleted to make room for new ones; thus, except for the first few iterations, we always have the $\hat{m}_c$ most recent correction pairs $(\boldsymbol{s}_i, \boldsymbol{u}_i)$ available.

Similarly to [3] we define the limited memory BFGS update by the formula

$$D_k = \vartheta_k I + \begin{bmatrix} S_k & \vartheta_k U_k \end{bmatrix} \begin{bmatrix} (R_k^{-1})^T (C_k + \vartheta_k U_k^T U_k) R_k^{-1} & -(R_k^{-1})^T \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^T \\ \vartheta_k U_k^T \end{bmatrix}. \tag{14}$$

Here, R_k is an upper triangular matrix of order $\hat{m}_k$ given by the form

$$(R_k)_{ij} = \begin{cases} (\boldsymbol{s}_{k-\hat{m}_k-1+i})^T (\boldsymbol{u}_{k-\hat{m}_k-1+j}), & \text{if } i \leq j \\ 0, & \text{otherwise,} \end{cases}$$

C_k is a diagonal matrix of order $\hat{m}_k$ such that

$$C_k = \operatorname{diag}\left[\boldsymbol{s}_{k-\hat{m}_k}^T \boldsymbol{u}_{k-\hat{m}_k}, \dots, \boldsymbol{s}_{k-1}^T \boldsymbol{u}_{k-1}\right],$$

and the scaling parameter $\vartheta_k > 0$ is given by

$$\vartheta_k = \frac{\boldsymbol{u}_{k-1}^T \boldsymbol{s}_{k-1}}{\boldsymbol{u}_{k-1}^T \boldsymbol{u}_{k-1}}. \tag{15}$$

In addition, we define the limited memory SR1 update (see, e.g., [3]) by

$$D_k = \vartheta_k I - (\vartheta_k U_k - S_k)(\vartheta_k U_k^T U_k - R_k - R_k^T + C_k)^{-1} (\vartheta_k U_k - S_k)^T, \tag{16}$$

where instead of (15) we use the value $\vartheta_k = 1$ for every k.

Next, we describe some procedures for updating the limited memory matrices. As said before, we use the limited memory BFGS update (14) to calculate the search direction $\boldsymbol{d}_k = -D_k\tilde{\boldsymbol{\xi}}_k$ if the previous step was a serious step. Now, after a serious step, the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k \in \partial f(\boldsymbol{x}_k)$ and the correction vectors obtained at the previous iteration (in Step 5 of Algorithm 1) can be equally expressed as $\boldsymbol{s}_{k-1} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ and $\boldsymbol{u}_{k-1} = \boldsymbol{\xi}_k - \boldsymbol{\xi}_{k-1}$. Therefore, the calculation of the search direction is very similar to that given in [3]. In fact, all the calculations in [3] can be done by replacing the gradient $\nabla f(\boldsymbol{x})$ by an arbitrary subgradient $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$. For more details of the BFGS update and its usage in limited memory bundle method we refer to [10,11].

After a null step, the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$ is not equal to $\boldsymbol{\xi}_k \in \partial f(\boldsymbol{y}_k)$ and the search direction $\boldsymbol{d}_k = -D_k\tilde{\boldsymbol{\xi}}_k$ is calculated using the limited memory SR1 update (16). In order to guarantee the global convergence of the method, the sequence (w_k) (see (7)) has to be nonincreasing in consecutive null steps (that is, $w_k \leq w_{k-1}$ if $i_{null} = k - m > 1$). Therefore, the condition

$$\tilde{\boldsymbol{\xi}}_k^T (D_k - D_{k-1})\tilde{\boldsymbol{\xi}}_k \leq 0 \tag{17}$$

has to be satisfied each time there occurs more than one consecutive null step (see the proof of Lemma 7).

We now give an efficient algorithm for direction finding using the limited memory SR1 update (16). This algorithm is used whenever the previous step was a null step. Together with Step 3 of Algorithm 1, this procedure guarantees that condition (17) is valid even when the correction (6) is executed (see Lemma 6). To simplify the notation, we now, for a while, omit the index $(k-1)$ and replace the index k by "+".

Algorithm 3 *(SR1 Updating and Direction Finding).*

Data: Suppose that the number of current correction pairs is $\hat{m}$ and the maximum number of stored correction pairs is $\hat{m}_c$. Suppose that we have the most recent vectors $\boldsymbol{s}$ and $\boldsymbol{u}$ (from the previous iteration), the current aggregate subgradient $\tilde{\boldsymbol{\xi}}_+$, the previous aggregate subgradient $\tilde{\boldsymbol{\xi}}$, the previous search direction $\boldsymbol{d}$, the $n \times \hat{m}$ matrices S and U, the $\hat{m} \times \hat{m}$ matrices R, U^TU, and C, and the previous scaling parameter ϑ available. In addition, suppose that we have the number of consecutive null steps $i_{null} \geq 1$.

Step 1: (*Initial vectors and initialization.*) Compute $\hat{m}$-vectors $S^T\tilde{\boldsymbol{\xi}}_+$ and $U^T\tilde{\boldsymbol{\xi}}_+$. Set $\vartheta_+ = 1.0$ and $i_{up} = 0$.

Step 2: (*Positive definiteness.*) If

$$-\boldsymbol{d}^T\boldsymbol{u} - \tilde{\boldsymbol{\xi}}^T\boldsymbol{s} < 0, \tag{18}$$

then set $\hat{m}_+ = \min\{\hat{m}+1, \hat{m}_c\}$ and calculate $\boldsymbol{s}^T\tilde{\boldsymbol{\xi}}_+$ and $\boldsymbol{u}^T\tilde{\boldsymbol{\xi}}_+$. Otherwise, skip the updates. That is, set $S_+ = S$, $U_+ = U$, $R_+ = R$, $U_+^TU_+ = U^TU$, $C_+ = C$, $S_+^T\tilde{\boldsymbol{\xi}}_+ = S^T\tilde{\boldsymbol{\xi}}_+$, $U_+^T\tilde{\boldsymbol{\xi}}_+ = U^T\tilde{\boldsymbol{\xi}}_+$, and $\hat{m}_+ = \hat{m}$ and go to Step 8.

Step 3: (*Update conditions.*) If either

$$i_{null} = 1 \qquad \text{or} \qquad \hat{m}_+ < \hat{m}_c,$$

then update the matrices, that is, go to Step 4. Otherwise, solve the system of linear equations

$$(\vartheta U^T U - R - R^T + C)\boldsymbol{p} = \vartheta U^T \tilde{\boldsymbol{\xi}}_+ - S^T \tilde{\boldsymbol{\xi}}_+.$$

to obtain $\boldsymbol{p} \in \mathbb{R}^{\hat{m}}$. Calculate the vector $\boldsymbol{z} \in \mathbb{R}^n$ as

$$\boldsymbol{z} = \vartheta \tilde{\boldsymbol{\xi}}_+ - (\vartheta U - S)\boldsymbol{p},$$

and the scalar

$$a = \tilde{\boldsymbol{\xi}}_+^T \boldsymbol{z}.$$

Set $i_{up} = 1$.

Step 4: Obtain S_+ and U_+ by updating S and U.

Step 5: Compute $\hat{m}_+$-vectors $S_+^T \boldsymbol{u}$ and $U_+^T \boldsymbol{u}$.

Step 6: Update $\hat{m}_+ \times \hat{m}_+$ matrices R_+, $U_+^T U_+$, and C_+.

Step 7: Construct $\hat{m}_+$-vectors $S_+^T \tilde{\boldsymbol{\xi}}_+$ and $U_+^T \tilde{\boldsymbol{\xi}}_+$ using $S^T \tilde{\boldsymbol{\xi}}_+$, $U^T \tilde{\boldsymbol{\xi}}_+$, $\boldsymbol{s}^T \tilde{\boldsymbol{\xi}}_+$, and $\boldsymbol{u}^T \tilde{\boldsymbol{\xi}}_+$.

Step 8: (*Intermediate value.*) Solve $\boldsymbol{p} \in \mathbb{R}^{\hat{m}_+}$ satisfying the system of linear equations

$$(\vartheta_+ U_+^T U_+ - R_+ - R_+^T + C_+)\boldsymbol{p} = \vartheta_+ U_+^T \tilde{\boldsymbol{\xi}}_+ - S_+^T \tilde{\boldsymbol{\xi}}_+.$$

Step 9: (*Search direction.*) Compute

$$\boldsymbol{d}_+ = -\vartheta_+ \tilde{\boldsymbol{\xi}}_+ + (\vartheta_+ U_+ - S_+)\boldsymbol{p}.$$

Step 10: (*Update conditions II.*) If $i_{up} = 1$, then calculate

$$b = \tilde{\boldsymbol{\xi}}_+ \boldsymbol{d}_+,$$

and in case of

$$b + a < 0,$$

(i.e., condition (17) is not valid) set $\hat{m}_+ = \hat{m}$, $S_+ = S$, $U_+ = U$, $R_+ = R$, $U_+^T U_+ = U^T U$, $C_+ = C$, and

$$\boldsymbol{d}_+ = -\boldsymbol{z}.$$

Condition (18) assures the positive definiteness of the matrices obtained by the limited memory SR1 update (see Appendix). Moreover, it implies $\boldsymbol{u}_{k-1}^T \boldsymbol{s}_{k-1} > 0$ that assures the positive definiteness of the matrices obtained by the limited memory BFGS update (see, e.g., [3]). Since we check condition (18) also during the limited memory BFGS update before updating matrices (see [10,11]), all the matrices D_k formed in the limited memory bundle method are positive definite.

3 Convergence Analysis

In this section, we prove the global convergence of Algorithm 1. The necessary condition for a locally Lipschitz continuous objective function to attain its local minimum (in an unconstrained case) is that $\mathbf{0} \in \partial f(\boldsymbol{x})$ (i.e., $\boldsymbol{x}$ is a stationary point, see, e.g., [4]). For a convex function this condition is also sufficient and the minimum is global.

Now, in addition to assuming that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous, the level set $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ is supposed to be bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Furthermore, we assume that each execution of the line search procedure is finite (that is, the objective function is assumed to be semi-smooth [2]). Since the optimality condition $\mathbf{0} \in \partial f(\boldsymbol{x})$ is not sufficient without some convexity assumptions, and the objective function f is not supposed to be convex, we can only prove that Algorithm 1 either terminates at a stationary point or generates an infinite sequence $(\boldsymbol{x}_k)$ for which accumulation points are stationary for f. In order to do this, we assume that the final accuracy tolerance ε is equal to zero.

We start the convergence analysis by giving three technical results (Lemmas 1, 2, and 3). After that, we prove (in Theorem 1) that having the values $w_k = 0$ and $q_k = 0$ implies that the corresponding point $\boldsymbol{x}_k$ is a stationary point for the objective function. For an infinite sequence $(\boldsymbol{x}_k)$, we first show (in Lemma 4) that the conditions $(q_k) \to 0$ and $(w_k) \to 0$ are equivalent due to correction (6) and, thus, we can restrict our consideration to the stopping parameter w_k. Then we show (in Lemma 5) that if $(\boldsymbol{x}_k)_{k \in \mathcal{K}} \to \bar{\boldsymbol{x}}$ and $(w_k)_{k \in \mathcal{K}} \to 0$ for some subset $\mathcal{K} \subset \{1, 2, \ldots\}$, then the accumulation point $\bar{\boldsymbol{x}}$ is a stationary point for the objective function. This assertion requires the uniformly positive definiteness of D_k, which also is guaranteed by correction (6). Furthermore, using the fact that the sequence (w_k) is nonincreasing in the consecutive null steps due to an additional testing procedure during the limited memory SR1 update (see Lemma 6), we prove that the indefinite sequence of consecutive null steps with $\boldsymbol{x}_k = \boldsymbol{x}_m$ implies $\mathbf{0} \in \partial f(\boldsymbol{x}_m)$ (in Lemma 7). Finally, in Theorem 2 we combine all the results obtained and show that every accumulation point of $(\boldsymbol{x}_k)$ is stationary for the objective function.

In principle, the convergence analysis of the limited memory bundle method reminds that of the original variable metric bundle method for nonconvex objective functions [34]. In fact, Lemmas 2 and 3 (with their proofs) are exactly the same as those given in [34]. In addition, Lemmas 5 and 7 as well as Theorems 1 and 2 are similar to the corresponding ones in [34] but their proofs have to be adopted. The main differences between the convergence analysis of the limited memory bundle method and the variable metric bundle method [34] follow from the facts that in the limited memory bundle method we have two different stopping parameters w_k and q_k (see (7) and (8)) instead of only one and, moreover, we are not able to guarantee that the sequence (w_k) is nonincreasing in the consecutive null steps without an additional testing procedure during the limited memory SR1 update (Algorithm 3). Nevertheless, in Lemma 4, we show that $(q_k) \to 0$ implies $(w_k) \to 0$ and vice versa

and, thus, it is enough to examine only the stopping parameter w_k, which is similar to that used in [34]. Furthermore, in Lemma 6, we first show that along with Step 3 of Algorithm 1, the procedure used in Algorithm 3 guarantees that condition (17) is valid even if the correction (6) is employed and then, in the proof of Lemma 7, we show that having condition (17) satisfied guarantees that the sequence (w_k) is nonincreasing.

Lemma 1 *At the kth iteration of Algorithm 1, we have*

$$\begin{aligned} w_k &= \tilde{\boldsymbol{\xi}}_k^T D_k \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \varrho \|\tilde{\boldsymbol{\xi}}_k\|^2, \\ q_k &= \frac{1}{2}\|\tilde{\boldsymbol{\xi}}_k\|^2 + \tilde{\beta}_k, \qquad q_k \geq \tilde{\beta}_k, \qquad q_k \geq \frac{1}{2}\|\tilde{\boldsymbol{\xi}}_k\|^2, \end{aligned}$$

and

$$\beta_{k+1} \geq \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^{\omega}. \tag{19}$$

Furthermore, if condition (18) is valid for $k = k + 1$, then

$$\boldsymbol{u}_k^T (D_k \boldsymbol{u}_k - \boldsymbol{s}_k) > 0. \tag{20}$$

Proof We point out first that $\tilde{\beta}_k \geq 0$ for all k by (9), (12), and Step 1 in Algorithm 1. The relations

$$\begin{aligned} w_k &= \tilde{\boldsymbol{\xi}}_k^T D_k \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \varrho \|\tilde{\boldsymbol{\xi}}_k\|^2, \\ q_k &= \frac{1}{2}\|\tilde{\boldsymbol{\xi}}_k\|^2 + \tilde{\beta}_k, \qquad q_k \geq \tilde{\beta}_k, \qquad q_k \geq \frac{1}{2}\|\tilde{\boldsymbol{\xi}}_k\|^2 \end{aligned}$$

follow immediately from (5), (6), (7), and (8). Note that, if correction (6) is used, we have $D_k = D_k + \varrho I$ and, thus, these results are valid also in this case.

By (9) and since we have $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps, and since we, on the other hand, have $\beta_{k+1} = 0$ and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| = 0$ for serious steps, condition (19) always holds for some $\gamma \geq 0$ and $\omega \geq 1$.

Now, we prove that condition (18) implies (20). If condition (18) is valid for k replaced with $k + 1$, then $\tilde{\boldsymbol{\xi}_k} \neq \mathbf{0}$ (otherwise, we would have $-\boldsymbol{d}_k^T \boldsymbol{u}_k - \tilde{\boldsymbol{\xi}}_k^T \boldsymbol{s}_k = 0$). Furthermore, we have

$$\boldsymbol{d}_k^T \boldsymbol{u}_k > -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{s}_k = t_R^k \theta_k \tilde{\boldsymbol{\xi}}_k^T D_k \tilde{\boldsymbol{\xi}}_k, \tag{21}$$

with $t_R^k > 0$ and $\theta_k \in (0, 1]$. Using the positiveness of $\boldsymbol{u}_k^T \boldsymbol{s}_k$ (provided by the positive definiteness of D_k in (21)), Cauchy's inequality, and the fact that $\boldsymbol{s}_k = t_R^k \theta_k \boldsymbol{d}_k$ we obtain

$$\begin{aligned} (\boldsymbol{u}_k^T \boldsymbol{s}_k)^2 &= (t_R^k \theta_k \tilde{\boldsymbol{\xi}}_k^T D_k \boldsymbol{u}_k)^2 \\ &\leq (t_R^k \theta_k)^2 \tilde{\boldsymbol{\xi}}_k^T D_k \tilde{\boldsymbol{\xi}}_k \boldsymbol{u}_k^T D_k \boldsymbol{u}_k \\ &= t_R^k \theta_k \boldsymbol{u}_k^T D_k \boldsymbol{u}_k (-\boldsymbol{s}_k^T \tilde{\boldsymbol{\xi}}_k) \\ &< t_R^k \theta_k \boldsymbol{u}_k^T D_k \boldsymbol{u}_k \boldsymbol{d}_k^T \boldsymbol{u}_k = \boldsymbol{u}_k^T D_k \boldsymbol{u}_k \boldsymbol{u}_k^T \boldsymbol{s}_k. \end{aligned}$$

Therefore, we have $\boldsymbol{u}_k^T \boldsymbol{s}_k < \boldsymbol{u}_k^T D_k \boldsymbol{u}_k$. □

Lemma 2 *Suppose that Algorithm 1 is not terminated before the kth iteration. Then, there exist numbers* $\lambda^{k,j} \geq 0$ *for* $j = 1, \ldots, k$ *and* $\tilde{\sigma}_k \geq 0$ *such that*

$$(\tilde{\boldsymbol{\xi}}_k, \tilde{\sigma}_k) = \sum_{j=1}^{k} \lambda^{k,j} (\boldsymbol{\xi}_j, \|\boldsymbol{y}_j - \boldsymbol{x}_k\|), \qquad \sum_{j=1}^{k} \lambda^{k,j} = 1, \quad \textit{and} \quad \tilde{\beta}_k \geq \gamma \tilde{\sigma}_k^{\omega}.$$

Proof See the proof of Lemma 3.2 in [34]. □

Lemma 3 *Let* $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ *be given and suppose that there exist vectors* $\bar{\boldsymbol{g}}$, $\bar{\boldsymbol{\xi}}_i$, $\bar{\boldsymbol{y}}_i$, *and numbers* $\bar{\lambda}_i \geq 0$ *for* $i = 1, \ldots, l$, $l \geq 1$, *such that*

$$\begin{aligned} &(\bar{\boldsymbol{g}}, 0) = \sum_{i=1}^{l} \bar{\lambda}_i (\bar{\boldsymbol{\xi}}_i, \|\bar{\boldsymbol{y}}_i - \bar{\boldsymbol{x}}\|), \\ &\bar{\boldsymbol{\xi}}_i \in \partial f(\bar{\boldsymbol{y}}_i), \quad i = 1, \ldots, l, \qquad \textit{and} \\ &\sum_{i=1}^{l} \bar{\lambda}_i = 1. \end{aligned}$$

Then $\bar{\boldsymbol{g}} \in \partial f(\bar{\boldsymbol{x}})$.

Proof See the proof of Lemma 3.3 in [34]. □

Theorem 1 *If Algorithm 1 terminates at the kth iteration, then the point* $\boldsymbol{x}_k$ *is stationary for* f.

Proof If Algorithm 1 terminates at Step 4, then the fact $\varepsilon = 0$ implies that $w_k = 0$ and $q_k = 0$. Thus, $\tilde{\boldsymbol{\xi}}_k = \mathbf{0}$ and $\tilde{\beta}_k = \tilde{\sigma}_k = 0$ by Lemma 1 and Lemma 2.

Now, by Lemma 2 and by using Lemma 3 with

$$\begin{aligned} &\bar{\boldsymbol{x}} = \boldsymbol{x}_k, \qquad l = k, \qquad \bar{\boldsymbol{g}} = \tilde{\boldsymbol{\xi}}_k, \\ &\bar{\boldsymbol{\xi}}_i = \boldsymbol{\xi}_i, \qquad \bar{\boldsymbol{y}}_i = \boldsymbol{y}_i, \quad \bar{\lambda}_i = \lambda^{k,i} \quad \text{for } i \leq k, \end{aligned}$$

we obtain $\mathbf{0} = \tilde{\boldsymbol{\xi}}_k \in \partial f(\boldsymbol{x}_k)$ and, thus, $\boldsymbol{x}_k$ is stationary for f. □

From now on, we suppose that Algorithm 1 does not terminate, that is, $w_k > 0$ and $q_k > 0$ for all k.

Lemma 4 *Let the stopping parameters* w_k *and* q_k *of Algorithm 1 be defined by (7) and (8), respectively. Then*

$$(q_k) \to 0 \qquad \textit{if and only if} \qquad (w_k) \to 0.$$

Proof The condition $(q_k) \to 0$ implies $(\tilde{\boldsymbol{\xi}}_k) \to \mathbf{0}$ and $(\tilde{\beta}_k) \to 0$ by Lemma 1 and, thus, we have $(w_k) \to 0$.

On the other hand, by Lemma 1 we have $w_k \geq 2\tilde{\beta}_k \geq 0$ and $w_k \geq \varrho \|\tilde{\boldsymbol{\xi}}_k\|^2$ for some correction parameter $\varrho \in (0, 1/2)$. Therefore, $(w_k) \to 0$ implies $(\tilde{\beta}_k) \to 0$ and $(\tilde{\boldsymbol{\xi}}_k) \to \mathbf{0}$ and, thus, also $(q_k) \to 0$. □

In view of Lemma 4 we can, from now on, restrict our consideration to the stopping parameter w_k.

Lemma 5 *Suppose that the level set* $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ *is bounded. Then, the sequences* $(\boldsymbol{y}_k)$ *and* $(\boldsymbol{\xi}_k)$ *are also bounded. If, in addition, there exist a point* $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ *and an infinite set* $\mathcal{K} \subset \{1, 2, \ldots\}$ *such that* $(\boldsymbol{x}_k)_{k \in \mathcal{K}} \to \bar{\boldsymbol{x}}$ *and* $(w_k)_{k \in \mathcal{K}} \to 0$*, then* $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$.

Proof By noticing that the sequence $(\boldsymbol{y}_k)$ is bounded, since $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ for serious steps and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| \leq t_{max} C$ for null steps (by (2) and due to the fact that we use the scaled direction vector $\theta_k \boldsymbol{d}_k$ with $\theta_k = \min\{1, C/\|\boldsymbol{d}_k\|\}$ and predefined $C > 0$ in the line search), the proof is similar to the proof of Lemma 3.4 in [34]. □

Lemma 6 *Suppose that the level set* $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ *is bounded, the number of serious steps is finite, and the last serious step occurred at the iteration* $m - 1$*. Then there exists a number* $k^* \geq m$*, such that*

$$\tilde{\boldsymbol{\xi}}_{k+1}^T D_{k+1} \tilde{\boldsymbol{\xi}}_{k+1} \leq \tilde{\boldsymbol{\xi}}_{k+1}^T D_k \tilde{\boldsymbol{\xi}}_{k+1} \qquad \textit{and} \tag{22}$$

$$\operatorname{tr}(D_k) < \frac{3}{2} n \tag{23}$$

for all $k \geq k^*$*, where* $\operatorname{tr}(D_k)$ *denotes the trace of matrix* D_k.

Proof Suppose first that $i_{CN} = 0$ for all $k \geq m$, that is, the correction ϱI (see Algorithm 1, Step 3) is not added to any matrix D_k with $k \geq m$. If the limited memory SR1 update is not used, we have $D_{k+1} = D_k$, and condition (22) is valid with equalities. Otherwise, if $\hat{m}_k < \hat{m}_c$, the limited memory SR1 update is equal to the standard SR1 update (see, e.g., [7]) and we have

$$D_{k+1} = D_k - \frac{(D_k \boldsymbol{u}_k - \boldsymbol{s}_k)(D_k \boldsymbol{u}_k - \boldsymbol{s}_k)^T}{\boldsymbol{u}_k^T (D_k \boldsymbol{u}_k - \boldsymbol{s}_k)},$$

where for the denominator we have $\boldsymbol{u}_k^T (D_k \boldsymbol{u}_k - \boldsymbol{s}_k) > 0$ by Lemma 1 and the numerator is positive (semi)definite matrix. Thus, condition (22) is valid in the case $\hat{m}_k < \hat{m}_c$. Finally, if $\hat{m}_k = \hat{m}_c$, we update the matrix only if $\tilde{\boldsymbol{\xi}}_{k+1}^T (D_{k+1} - D_k) \tilde{\boldsymbol{\xi}}_{k+1} \leq 0$ (see Algorithm 3, Steps 3 and 10). Since $i_{CN} = 0$ for all $k \geq m$, the correction ϱI is not added to the new matrix D_{k+1} and, thus, condition (22) is valid also in this case. Furthermore, in each case we have

$$\operatorname{tr}(D_k) - \frac{3}{2} n = \operatorname{tr}(D_k) - \operatorname{tr}(I) - \frac{1}{2} n = \operatorname{tr}(D_k - I) - \frac{1}{2} n < 0$$

for $k \geq m$, since the matrix $D_k - I$ is negative (semi)definite due to condition (18). Therefore, if $i_{CN} = 0$ for all $k \geq m$, conditions (22) and (23) are valid if we set $k^* = m$.

If $i_{CN} = 0$ does not hold for all $k \geq m$, then the correction ϱI, with $\varrho \in (0, 1/2)$, is added to all the matrices D_k with $k \geq \bar{k}$ (see Algorithm 1, Step 3). Here $\bar{k}$ denotes the index $k \geq m$ of the iteration when $i_{CN} = 1$ occurred for the first time. The limited memory matrices S_k, U_k, R_k, and C_k

do not contain information of the correction ϱI that may have been added to the matrix D_k at the previous iteration. Let us now denote by $\hat{D}_k$ the matrix formed with the limited memory matrices and by D_k the corrected matrix, that is, $D_k = \hat{D}_k + \varrho I$ for all $k \geq \bar{k}$ (since we suppose $i_{CN} = 1$).

Now, all the results given above are valid for $\hat{D}_k$ and $\hat{D}_{k+1}$. Since for all $k \geq \bar{k}$, we have $D_k = \hat{D}_k + \varrho I$ and we set $D_{k+1} = \hat{D}_{k+1} + \varrho I$, condition (22) is valid for all $k \geq k^* = \bar{k}$. Now, in each case we have

$$\begin{aligned}
\mathrm{tr}(D_k) - \frac{3}{2}n &= \mathrm{tr}(\hat{D}_k + \varrho I) - \mathrm{tr}(I) - \frac{1}{2}n \\
&= \mathrm{tr}(\hat{D}_k - I) + \mathrm{tr}(\varrho I) - \frac{1}{2}n \\
&< \mathrm{tr}(\hat{D}_k - I) + \frac{1}{2}n - \frac{1}{2}n \\
&\leq 0
\end{aligned}$$

for $k \geq \bar{k}$, since the matrix $\hat{D}_k - I$ is negative (semi)definite and $\varrho \in (0, 1/2)$. Therefore, conditions (22) and (23) are valid for all $k \geq k^*$ with

$$k^* = \max\{\bar{k}, m\},$$

where $\bar{k} = 1$ if $i_{CN} = 0$. □

Lemma 7 *Suppose that the level set* $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ *is bounded, the number of serious steps is finite, and the last serious step occurred at the iteration* $m-1$*. Then, the point* $\boldsymbol{x}_m$ *is stationary for* f*.*

Proof From (10), (11), (12), Lemma 1, and Lemma 6 we obtain

$$\begin{aligned}
w_{k+1} &= \tilde{\boldsymbol{\xi}}_{k+1}^T D_{k+1} \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\
&\leq \tilde{\boldsymbol{\xi}}_{k+1}^T D_k \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\
&= \varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) \\
&\leq \varphi(0, 0, 1) \\
&= \tilde{\boldsymbol{\xi}}_k^T D_k \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k \\
&= w_k
\end{aligned} \tag{24}$$

for $k \geq k^*$ with k^* defined in Lemma 6.

Let us denote $D_k = W_k^T W_k$. Then, function φ (see (10)) can be given in the form

$$\varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) = \|\lambda_1^k W_k \boldsymbol{\xi}_m + \lambda_2^k W_k \boldsymbol{\xi}_{k+1} + \lambda_3^k W_k \tilde{\boldsymbol{\xi}}_k\|^2 + 2(\lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k).$$

From (24) we obtain the boundedness of the sequences (w_k), $(W_k \tilde{\boldsymbol{\xi}}_k)$, and $(\tilde{\beta}_k)$. Furthermore, Lemma 6 assures the boundedness of (D_k) and (W_k). By Lemma 5, we obtain the boundedness of $(\boldsymbol{y}_k)$, $(\boldsymbol{\xi}_k)$, and $(W_k \boldsymbol{\xi}_{k+1})$.

Now, by noticing that we use the scaled direction vector $\theta_k \boldsymbol{d}_k$ (with $\theta_k = \min\{1, C/\|\boldsymbol{d}_k\|\}$ and predefined $C > 0$) and the scaled line search parameter $\varepsilon_R^k = \theta_k \varepsilon_R^I$ in the line search (see Algorithm 1 Step 5 and Algorithm 2) the last part of the proof proceeds similar to the proof (part (ii)) of Lemma 3.6 in [34]. □

Theorem 2 *Suppose that the level set* $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ *is bounded. Then, every accumulation point of the sequence* $(\boldsymbol{x}_k)$ *is stationary for* f.

Proof Let $\bar{\boldsymbol{x}}$ be an accumulation point of $(\boldsymbol{x}_k)$, and let $\mathcal{K} \subset \{1, 2, \ldots\}$ be an infinite set such that $(\boldsymbol{x}_k)_{k \in \mathcal{K}} \to \bar{\boldsymbol{x}}$. In view of Lemma 7, we can restrict our consideration to the case where the number of serious steps (with $t_L^k > 0$) is infinite. We denote

$$\mathcal{K}' = \{k \mid t_L^k > 0, \text{ there exists } i \in \mathcal{K},\ i \leq k \text{ such that } \boldsymbol{x}_i = \boldsymbol{x}_k\}.$$

Obviously, $\mathcal{K}'$ is infinite and $(\boldsymbol{x}_k)_{k \in \mathcal{K}'} \to \bar{\boldsymbol{x}}$. The continuity of f implies that $(f_k)_{k \in \mathcal{K}'} \to f(\bar{\boldsymbol{x}})$ and, thus, $f_k \downarrow f(\bar{\boldsymbol{x}})$ by the monotonicity of the sequence (f_k) obtained due to the serious descent criterion (13). Using the fact that $t_L^k \geq 0$ for all $k \geq 1$ and condition (13), we obtain

$$0 \leq \varepsilon_L^k t_L^k w_k \leq f_k - f_{k+1} \to 0 \qquad \text{for } k \geq 1. \tag{25}$$

If the set $\mathcal{K}_1 = \{k \in \mathcal{K}' \mid t_L^k \geq t_{min}\}$ is infinite, then $(w_k)_{k \in \mathcal{K}_1} \to 0$ and $(\boldsymbol{x}_k)_{k \in \mathcal{K}_1} \to \bar{\boldsymbol{x}}$ by (25) and, thus, by Lemma 5 we have $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$.

If the set $\mathcal{K}_1$ is finite, then the set $\mathcal{K}_2 = \{k \in \mathcal{K}' \mid \beta_{k+1} > \varepsilon_A^k w_k\}$ has to be infinite (see Algorithm 2, Step 2). To the contrary, let us assume that

$$w_k \geq \delta > 0, \qquad \text{for all } k \in \mathcal{K}_2.$$

From (25), we have $(t_L^k)_{k \in \mathcal{K}_2} \to 0$ and Step 5 in Algorithm 1 implies

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| = t_L^k \theta_k \|\boldsymbol{d}_k\| \leq t_L^k C$$

for all $k \geq 1$. Thus, we have $(\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|)_{k \in \mathcal{K}_2} \to 0$. By (9), (25), and the boundedness of $(\boldsymbol{\xi}_k)$ by Lemma 5, and since $\boldsymbol{y}_{k+1} = \boldsymbol{x}_{k+1}$ for serious steps, we obtain $(\beta_{k+1})_{k \in \mathcal{K}_2} \to 0$, which is in contradiction with

$$\varepsilon_A^k \delta \leq \varepsilon_A^k w_k < \beta_{k+1}, \qquad k \in \mathcal{K}_2.$$

Therefore, there exists an infinite set $\mathcal{K}_3 \subset \mathcal{K}_2$ such that $(w_k)_{k \in \mathcal{K}_3} \to 0$, $(\boldsymbol{x}_k)_{k \in \mathcal{K}_3} \to \bar{\boldsymbol{x}}$, and $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$ by Lemma 5. □

Note that, if we choose $\varepsilon > 0$, Algorithm 1 terminates in a finite number of steps. This is due to fact that if the number of serious steps is finite we have $(w_k) \to 0$ and $(q_k) \to 0$ (see Lemma 4, the proof of Lemma 7, and the proof (part (ii)) of Lemma 3.6 in [34]), and if the number of serious steps is infinite we have either $(w_k)_{k \in \mathcal{K}_1} \to 0$ and $(q_k)_{k \in \mathcal{K}_1} \to 0$ or $(w_k)_{k \in \mathcal{K}_3} \to 0$ and $(q_k)_{k \in \mathcal{K}_3} \to 0$ (see Lemma 4 and the proof of Theorem 2).

4 Numerical Experiments

In this section we compare the limited memory bundle method (LMBM) to the proximal bundle method (PBNCGC) using both academic test problems and practical applications arising in the field of nonsmooth large-scale optimization. We use the solver PBNCGC as a benchmark since the proximal bundle method is the most frequently used bundle method in nonsmooth optimization. A more extensive numerical analysis concerning the performance of the limited memory bundle method and some other existing bundle methods can be found in [11].

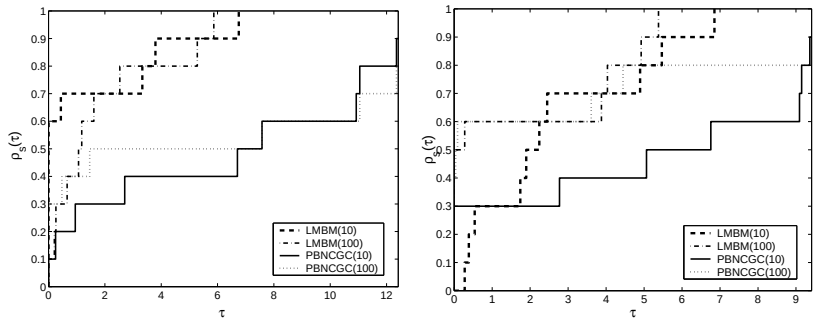
The experiments were mostly performed on a SGI Origin 2000/128 supercomputer (MIPS R12000, 600 Mflop/s/processor). However, for the last practical application (the hemivariational inequality problem [23]) the test runs were performed on an HP9000/J5600 workstation (2×552MHz, PA8600 CPU) because of libraries required. The tested algorithms were implemented in Fortran77 with double-precision arithmetic.

The tested proximal bundle algorithm PBNCGC is from the software package NSOLIB (NonSmooth Optimization LIBrary, see [21]). The solver utilizes the subgradient aggregation strategy of [15] and it uses the quadratic solver QPDF4 based on the dual active-set method [16] to solve the quadratic direction finding problem. For a detailed description of the solver see [21,24].

Nonsmooth academic problems. The solvers were first tested with 10 nonsmooth academic minimization problems described in [11]. Half of these problems were convex and the rest were nonconvex. The number of variables was 1000, and the solvers were tested with relatively small sizes of the bundles, that is, $m_\xi = 10$ and $m_\xi = 100$. In what follows, we denote these different variants by LMBM(10), LMBM(100), PBNCGC(10), and PBNCGC(100). For LMBM, the maximum number of stored corrections ($\hat{m}_c$) was set to 7 and the maximum number of additional interpolations used at each iteration was set to 200 (default value). For convex problems, we used the distance measure parameter $\gamma = 0$ and for nonconvex problems, we used the value $\gamma = 0.5$ with both the solvers. The stopping parameter $\varepsilon = 10^{-5}$ was used in all the cases. Otherwise, the default parameters of the solvers were used.

In the test results to be reported, we say that the optimization terminated successfully if the problem was solved with the desired accuracy. That is, $w_k \leq \varepsilon$, where w_k is defined by (7) for LMBM (note that also $q_k \leq \varepsilon$, see (8)) and $w_k = \frac{1}{2}\|\tilde{\boldsymbol{\xi}}_a^k\| + \tilde{\beta}_a^k$ for PBNCGC (here $\tilde{\boldsymbol{\xi}}_a^k$ is the aggregate subgradient and $\tilde{\beta}_a^k$ is the aggregate subgradient locality measure, see [21,24]). In addition, for LMBM, we say that the optimization terminated successfully if $|f_{k+1} - f_k| \leq 1.0 \cdot 10^{-8}$ in 10 subsequent iterations and the result obtained was less than two significant digits greater than the desired accuracy of the solution. In proportion, we accepted the result for PBNCGC if $|w_{k+1} - w_k| \leq 1.0 \cdot 10^{-12}$ in 10 subsequent iterations. Otherwise, we say that the optimization failed.

The results of the experiments are summarized in Figure 2. The test results are analyzed using the performance profiles introduced in [6]. As performance measures, we use computation times (in Figure 2(a)) and numbers of function evaluations (in Figure 2(b)). There are two important facts to be



(a) CPU time (b) Function evaluations

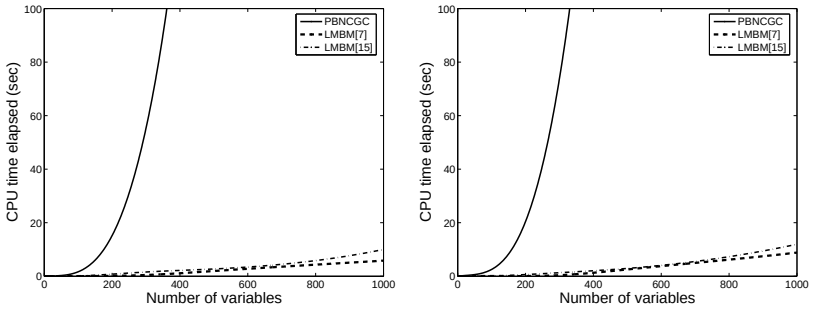
Fig. 2 Nonsmooth problems with 1000 variables.

kept in mind to have a good interpretation of the performance profiles. The value of $\rho(\tau)$ at $\tau = 0$ gives the percentage of test problems for which the corresponding solver is the best and the value of $\rho(\tau)$ at the rightmost abscissa is the percentage of test problems that the corresponding solver can solve (this does not depend on the measured performance). Finally, the relative efficiency of each solver can be directly seen from the performance profiles: the higher is the particular curve, the better is the corresponding solver. For more information of performance profiles, see [6].

In Figure 2(a) we see that `LMBM(10)` was the most efficient solver on 60% of the problems. It also solved the rest of the problems really fast while, with the proximal bundle solver `PBNCGC`, (with both sizes of bundles) there was a great variation in the computation times of different problems. The solver `LMBM(100)` was very efficient with these large-scale problems as well, although, it usually needed slightly more computation time than `LMBM(10)`. Besides being very efficient, `LMBM` succeed to solve all the problems while `PBNCGC` failed in some of the problems. The computation time elapsed with `LMBM` was on the average about 40 times shorter than that of `PBNCGC`.

The numbers of function evaluations used with `LMBM(100)` and `PBNCGC(100)` were approximately the same (see Figure 2(b)). With the solver `PBNCGC` the required number of function evaluations was significantly larger when the size of the bundle was small (that is, $m_\xi = 10$). This was usually true also for the solver `LMBM` but with `LMBM` the difference was not so substantial. With `LMBM` the decrease in the number of function evaluations when increasing the size of the bundle is due to the fact that the selection of the initial step size is more accurate when a larger bundle is used. In practice, this means that for problems with expensive objective function and subgradient evaluations, it is better to use larger bundles and, thus, fewer iterations and function evaluations.

Image restoration problems. In addition to the academic test problems, we tested `LMBM` and `PBNCGC` with two convex image restoration problems [13,14],



(a) Problem (P1) (b) Problem (P2)

Fig. 3 CPU times elapsed for the image restoration problems.

which are typical nonsmooth large-scale optimization problems arising in optimal control applications. In what follows, we denote by (P1) the problem described in [14] and by (P2) the problem described in [13]. Both of the problems are so-called noise reduction problems. That is, it is assumed that the observed images are degraded by a random noise. The noise reduction problems are formulated as minimization problems consisting of a least squares fit and a nonsmooth bounded variational type regularization term (for more details of the formulations, see [13,14]).

The solvers were tested with $m_\xi = 10$ for `LMBM` and $m_\xi = 100$ for `PBNCGC` (since the previous experiments have shown that a larger bundle usually works better with `PBNCGC`). The stopping parameter $\varepsilon = 10^{-4}$ was used with both the solvers and the value of the distance measure parameter γ was set to zero (since the objective functions are convex). We tested `LMBM` with different maximum numbers of stored correction pairs, that is, $\hat{m}_c = 7$ and $\hat{m}_c = 15$. In what follows, we denote these different variants by `LMBM[7]` and `LMBM[15]`, respectively. Otherwise, the default parameters of the solvers were used.

In Figure 3, we give the computation times elapsed for the problems with different numbers of variables (up to 1000 variables). Furthermore, we give some more specific results for the problems with 100, 300, 1000, and, in the case of `LMBM`, also with 10 000 variables in Tables 1, 2, and 3, where Ni and Nf denote the numbers of iterations and function evaluations used, respectively, f denotes the value of the objective function at termination, and CPU time (in Table 3) is given in seconds. In addition to the results obtained in our experiment, we give (in Table 1) the reported final values of the objective function for problem (P1) obtained with the special active set method (`ACS`) [14]. The active set method has not been applied for problem (P2) in [13], nor has it been applied for problems with 10 000 variables. Thus, in Tables 2 and 3, we only give the results obtained in our experiments.

Based on the numerical results, we can conclude the superiority of the limited memory bundle solvers `LMBM[7]` and `LMBM[15]` when comparing the computation times: in all the cases, they used significantly less CPU time

Table 1 Results for the image restoration problem (P1).

Solver/n	100		300		1000	
	Ni/Nf	f	Ni/Nf	f	Ni/Nf	f
ACS	–	0.7981	–	2.7586	–	9.8950
PBNCGC	186/187	0.7976	1388/1389	2.7580	16777/16778	9.7617
LMBM[7]	442/669	0.7979	1707/1919	2.7666	6343/6373	9.8152
LMBM[15]	598/1451	0.7976	3980/5184	2.7632	8285/8508	9.8042

Table 2 Results for the image restoration problem (P2).

Solver/n	100		300		1000	
	Ni/Nf	f	Ni/Nf	f	Ni/Nf	f
PBNCGC	249/250	0.5973	1597/1598	2.2517	17383/17384	7.9571
LMBM[7]	764/1062	0.5974	1476/1597	2.2603	10028/10073	7.9958
LMBM[15]	640/1515	0.5973	3324/4535	2.2568	9871/10054	7.9944

Table 3 Results for the image restoration problems with 10 000 variables.

	Problem (P1)			Problem (P2)		
Solver	Ni/Nf	f	CPU	Ni/Nf	f	CPU
LMBM[7]	78185/78217	106.0738	882.16	38721/38729	86.4913	389.13
LMBM[15]	78827/78866	106.0118	1163.64	50903/50947	86.4086	750.25

than the proximal bundle solver PBNCGC (see Figure 3) and, unlike PBNCGC, they could successfully solve problems with 10 000 variables within a reasonable time (see Table 3). The solver PBNCGC did not reach the desired accuracy with 10 000 variables within 15 hours and the experiments were then terminated. Moreover, the number of iterations and function evaluations required with LMBM[7] and LMBM[15] did not grow as fast as those required with PBNCGC when the number of variables was increased. Indeed, with large numbers of variables, both variants of LMBM needed significantly less iterations and function evaluations than PBNCGC (see Tables 1 and 2).

With academic nonsmooth problems, the accuracy of the limited memory bundle solver LMBM (with both sizes of bundles) was comparable to that of the proximal bundle solver PBNCGC. However, with the image restoration problems, the minima found with LMBM[7] and LMBM[15] could be slightly greater than those of PBNCGC (see Tables 1 and 2). In all image restoration problems but one with 100 variables, the optimization with LMBM was terminated because the value of the objective function did not change. Thus, even if the value of stopping parameter ε was decreased the results obtained with LMBM would not become any smaller. Nevertheless, the results obtained with LMBM usually became more accurate when the maximum number of stored correction pairs was increased (see Tables 1–3). Moreover, in most cases, the results obtained for problem (P1) with both variants of LMBM were better

Table 4 Results for hemivariational inequalities.

Solver	Ni/Nf	f	CPU
PBNCGC	1955/4272	−0.01258375	190.36
LMBM[7]	678/1052	−0.01258534	2.29
LMBM[15]	387/452	−0.01258349	1.44

than those obtained with the active-set method ACS used in [14] (see Table 1), and both visually and with respect to the reconstruction error (that is, the average error between the true signal and the obtained result [13,14]) all the results of LMBM were comparable to the results of the proximal bundle solver PBNCGC (for both the problems). In fact, in all large-scale cases the reconstruction errors obtained with the limited memory bundle solvers were smaller than those obtained with the proximal bundle solver PBNCGC.

Hemivariational inequality problem. Finally, we tested the solvers with a hemivariational inequality problem [23] that is a demanding nonconvex nonsmooth real-world large-scale optimization problem and in essence very different from the image restoration problems. Hemivariational inequalities can be considered as generalizations of variational inequalities. The origin of these kind of problems is in nonsmooth mechanics of solids, especially in nonmonotone contact problems. Typically, hemivariational inequalities can be formulated as substationary point problems of the corresponding nonsmooth nonconvex energy functionals. For more details of the mathematical theory an applications of hemivariational inequalities in general, see [28,32].

Similarly to previous experiments, the solvers were tested with $m_\xi = 10$ for LMBM[7] and LMBM[15] and with $m_\xi = 100$ for PBNCGC. In all cases the stopping parameter $\varepsilon = 10^{-7}$ was used and the value of the distance measure parameter γ was set to 0.5 (since the objective function is nonconvex). Otherwise, the default parameters of the solvers were used. The number of variables in our test problem was 840 and the bound constraints were omitted. For more details of the formulation of the problem, see [23].

The results of the experiment are given in Table 4 and, again, the solvers LMBM[7] and LMBM[15] were superior when comparing the computation times. Moreover, in this problem also the accuracies of the limited memory bundle solvers LMBM[7] and LMBM[15] were as good as that of the proximal bundle solver PBNCGC.

We can conclude from these experiments that the limited memory bundle method was very efficient for large-scale nonsmooth optimization. For academic problems with 1000 variables, LMBM was on the average about 40 times faster than the proximal bundle solver PBNCGC. Moreover, LMBM found the (local) minimum in a reliable way for both convex and nonconvex optimization problems (see Figure 2). For the demanding image restoration problems and hemivariational inequalities the differences in the computation times were even more perceptible: for instance, for image restoration problems the limited memory bundle solver LMBM was about 50 times faster than

PBNCGC already with 300 variables (see Figure 3) and, for the first time, these problems were successfully solved with 10 000 variables.

5 Conclusions

In this paper, we have described a new variant of a limited memory bundle method for nonsmooth large-scale optimization. This method is intended to fill the gap, which exists in the field of demanding nonsmooth optimization with large numbers of variables. We have proved the global convergence of the method for locally Lipschitz continuous objective functions, which are not necessarily differentiable or convex.

The numerical experiments reported confirm that the limited memory bundle solver is efficient for both convex and nonconvex large-scale nonsmooth optimization problems. With large numbers of variables it used significantly less CPU time than the other solver tested and, indeed, in some cases, only the limited memory bundle solver was able to solve large-scale problems within a reasonable time.

Although the method described here has already proven useful, there is some further work required before the idea is complete. Possible areas of future development include the following: an adaptive version of the method, where the maximum number of stored correction vectors may change during the computation, alternative ways of scaling the updates, and constraint handling.

Acknowledgements We would like to thank Dr. Ladislav Lukšan and Dr. Jan Vlček for the permission to use and modify their variable metric bundle software to make the method suitable for large-scale optimization. Dr. Ladislav Lukšan deserves also special thanks for enlightening discussions and feedback concerning the limited memory bundle method.

This work was financially supported by the Academy of Finland grant #104641, the COMAS Graduate School of the University of Jyvskyl, and the University of the Witwatersrand. The computational resources used in the experiments were provided by CSC, the Finnish IT Center for Science.

Appendix

Lemma 8 *The condition (see Algorithm 3, Step 2)*

$$-\boldsymbol{d}_i^T \boldsymbol{u}_i - \tilde{\boldsymbol{\xi}}_i^T \boldsymbol{s}_i < 0 \qquad \text{for all } i = 1, \ldots, k-1, \tag{26}$$

assures the positive definiteness of the matrices obtained by the limited memory SR1 update.

Proof Let us denote $B_i = D_i^{-1}$ for all $i = 1, \ldots, k$. We now prove that each matrix B_k, $k \geq 1$ is positive definite when condition (26) is valid. Note that if B_k is positive definite, then also its inverse D_k is positive definite.

By applying the Sherman-Morrison-Woodbury formula (see, e.g., [7]) to (16) we obtain

$$B_k = B_1 + (U_k - B_1 S_k)(L_k + L_k^T + C_k - S_k^T B_1 S_k)^{-1}(U_k - B_1 S_k)^T, \quad (27)$$

where matrices S_k, U_k, and C_k are defined as before, $B_1 = D_1^{-1} = I$ is a positive definite initial matrix, and

$$(L_k)_{ij} = \begin{cases} (\boldsymbol{s}_{k-\hat{m}_k-1+i})^T(\boldsymbol{u}_{k-\hat{m}_k-1+j}) & \text{if } i > j \\ 0 & \text{otherwise.} \end{cases}$$

Let us denote

$$Q_k = \left[\boldsymbol{q}_{k-\hat{m}_k} \ldots \boldsymbol{q}_{k-1}\right] = U_k - B_1 S_k \quad \text{and}$$
$$N_k = L_k + L_k^T + C_k - S_k^T B_1 S_k$$

and let us assume that B_{k-1} is positive definite for some $k > 1$ and that condition (26) is valid for $i = 1, \ldots, k-1$. We prove that also the matrix B_k is positive definite. To simplify the notation, we, from now on, omit the index $(k-1)$ and replace the index k by "+". Thus, equation (27) can be rewritten in the form

$$\begin{aligned} B_+ &= B_1 + Q_+ N_+^{-1} Q_+^T \\ &= B' + \frac{(\boldsymbol{u} - B'\boldsymbol{s})\,(\boldsymbol{u} - B'\boldsymbol{s})^T}{\boldsymbol{s}^T\,(\boldsymbol{u} - B'\boldsymbol{s})}, \end{aligned} \quad (28)$$

where $B' = B_1 + Q' N'^{-1} Q'^T$, N' is formed by deleting the first row and the first column from N if $k > \hat{m}_c + 1$, and Q' is formed by deleting the first column from Q. If $k \leq \hat{m}_c + 1$, then $N' = N$ and $Q' = Q$.

It can be easily seen from (28) that the new matrix B_+ is positive definite if B' is positive definite and the denominator $\boldsymbol{s}^T(\boldsymbol{u} - B'\boldsymbol{s})$ is greater than zero. Therefore, we first prove that B' is positive definite. The current matrix B is positive definite by assumption and, due to condition (26), also $QN^{-1}Q^T$, N^{-1}, and N are positive definite. The positive definiteness of N implies the positive definiteness of N' as a submatrix of matrix N. Now, since N' is positive definite also N'^{-1}, $Q'N'^{-1}Q'^T$, and, thus, $B' = B_1 + Q'N'^{-1}Q'^T$ are positive definite.

Using condition (26) and the fact that we have $\boldsymbol{s}_i = t_R^i \theta_i \boldsymbol{d}_i$, $t_R^i > 0$, $\theta_i \in (0, 1]$, and $\boldsymbol{d}_i = -D_i \tilde{\boldsymbol{\xi}}_i$ for all $i = 1, \ldots, k-1$, we obtain

$$\boldsymbol{d}_i^T \boldsymbol{u}_i > t_R^i \theta_i \tilde{\boldsymbol{\xi}}_i^T D_i \tilde{\boldsymbol{\xi}}_i = t_R^i \theta_i \tilde{\boldsymbol{\xi}}_i^T D_i B_i D_i \tilde{\boldsymbol{\xi}}_i = t_R^i \theta_i \boldsymbol{d}_i^T B_i \boldsymbol{d}_i \geq t_R^i \theta_i \boldsymbol{d}_i^T B_i' \boldsymbol{d}_i \quad (29)$$

for all $i = 1, \ldots, k-1$. The last inequality in formula (29) follows from the fact that for $i \leq \hat{m}_c + 1$ the positive definite matrix B_i is equal to B_i' and for

$i > \hat{m}_c + 1$ it can be given in the form

$$\begin{aligned}
B_i &= B_1 + Q_i N_i^{-1} Q_i^T \\
&= B_1 + \begin{bmatrix} \boldsymbol{q}_- & Q_i' \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_-^T \boldsymbol{q}_- & \boldsymbol{q}_-^T S_i' \\ S_i'^T \boldsymbol{q}_- & N_i' \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{q}_-^T \\ Q_i'^T \end{bmatrix} \\
&= B_1 + \begin{bmatrix} \boldsymbol{q}_- & Q_i' \end{bmatrix} \begin{bmatrix} 1/\delta & -\boldsymbol{q}_-^T S_i' N_i'^{-1}/\delta \\ -N_i'^{-1} S_i'^T \boldsymbol{q}_-/\delta & N_i'^{-1} + N_i'^{-1} S_i'^T \boldsymbol{q}_- \boldsymbol{q}_-^T S_i' N_i'^{-1}/\delta \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_-^T \\ Q_i'^T \end{bmatrix} \\
&= B_i' + (\boldsymbol{q}_- - Q_i' N_i'^{-1} S_i'^T \boldsymbol{q}_-)(\boldsymbol{q}_- - Q_i' N_i'^{-1} S_i'^T \boldsymbol{q}_-)^T/\delta.
\end{aligned}$$

Here the subscript "$-$" denotes the index $i - \hat{m}_i - 1$, the matrix S_i' is formed by deleting the first column from S_i, and the denominator

$$\delta = \boldsymbol{s}_-^T \boldsymbol{q}_- - \boldsymbol{q}_-^T S_i' N_i'^{-1} S_i'^T \boldsymbol{q}_-$$

is greater than zero, since the matrix N_i^{-1} is positive definite (that is, the upper left term $1/\delta$ has to be greater than zero).

From (29) and by using again the fact that $\boldsymbol{s}_i = t_R^i \theta_i \boldsymbol{d}_i$ we obtain

$$\boldsymbol{s}_i^T (\boldsymbol{u}_i - B_i' \boldsymbol{s}_i) > 0$$

for all $i = 1, \ldots, k-1$. Thus, the denominator in formula (28) is greater than zero and the new matrix B_+ and its inverse D_+ are positive definite. □

References

1. Bacaud, L., Lemaréchal, C., Renaud, A., Sagastizábal, C.: Bundle methods in stochastic optimal power management: A disaggregated approach using preconditioners. Comput. Optim. Appl. **20**(3), 227–244 (2001)
2. Bihain, A.: Optimization of upper semidifferentiable functions. J. Optimization Theory Appl. **4**, 545–568 (1984)
3. Byrd, R.H., Nocedal, J., Schnabel, R.B.: Representations of quasi-Newton matrices and their use in limited memory methods. Math. Program. **63**, 129–156 (1994)
4. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley-Interscience, New York (1983)
5. Clarke, F.H., Ledyaev, Y.S., Stern, R.J., Wolenski, P.R.: Nonsmooth Analysis and Control Theory. Springer, New York (1998)
6. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
7. Fletcher, R.: Practical Methods of Optimization, 2nd edn. John Wiley and Sons, Chichester (1987)
8. Gilbert, J.C., Lemaréchal, C.: Some numerical experiments with variable-storage quasi-Newton algorithms. Math. Program. **45**, 407–435 (1989)
9. Gröwe-Kuska, N., Kiwiel, K.C., Nowak, M.P., Römisch, W., Wegner, I.: Power management in a hydro-thermal system under uncertainty by Lagrangian relaxation. In: C. Greengard, A. Ruszczynski (eds.) Decision Making under Uncertainty: Energy and Power, *IMA Volumes in Mathematics and its Applications*, vol. 128, pp. 39–70. Springer, New York (2002)
10. Haarala, M.: Large-scale nonsmooth optimization: Variable metric bundle method with limited memory. Ph.D. thesis, University of Jyväskylä, Department of Mathematical Information Technology (2004)

11. Haarala, M., Miettinen, K., Mäkelä, M.M.: New limited memory bundle method for large-scale nonsmooth optimization. Optim. Methods Softw. **19**(6), 673–692 (2004)
12. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II. Springer-Verlag, Berlin (1993)
13. Kärkkäinen, T., Majava, K., Mäkelä, M.M.: Comparison of formulations and solution methods for image restoration problems. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, B 14/2000 University of Jyväskylä, Jyväskylä (2000)
14. Kärkkäinen, T., Majava, K., Mäkelä, M.M.: Comparison of formulations and solution methods for image restoration problems. Inverse Probl. **17**(6), 1977–1995 (2001)
15. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Lecture Notes in Mathematics 1133. Springer-Verlag, Berlin (1985)
16. Kiwiel, K.C.: A method for solving certain quadratic programming problems arising in nonsmooth optimization. IMA J. Numer. Anal. **6**, 137–152 (1986)
17. Lemaréchal, C.: Nondifferentiable optimization. In: G.L. Nemhauser, A.H.G. Rinnooy Kan, M.J. Todd (eds.) Optimization, pp. 529–572. Elsevier North-Holland, Inc., New York (1989)
18. Lemaréchal, C., Strodiot, J.J., Bihain, A.: On a bundle algorithm for nonsmooth optimization. In: O.L. Mangasarian, R.R. Mayer, S.M. Robinson (eds.) Nonlinear Programming, pp. 285–281. Academic Press, New York (1981)
19. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**, 503–528 (1989)
20. Lukšan, L., Vlček, J.: Globally convergent variable metric method for convex nonsmooth unconstrained minimization. J. Optimization Theory Appl. **102**, 593–613 (1999)
21. Mäkelä, M.M.: Issues of implementing a Fortran subroutine package NSOLIB for nonsmooth optimization. Technical Report 5/1993, Department of Mathematics, Laboratory of Scientific Computing, University of Jyväskylä (1993)
22. Mäkelä, M.M.: Survey of bundle methods for nonsmooth optimization. Optim. Methods Softw. **17**(1), 1–29 (2002)
23. Mäkelä, M.M., Miettinen, M., Lukšan, L., Vlček, J.: Comparing nonsmooth nonconvex bundle methods in solving hemivariational inequalities. J. Glob. Optim. **14**, 117–135 (1999)
24. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific Publishing Co., Singapore (1992)
25. Mifflin, R.: A modification and an extension of Lemaréchal's algorithm for nonsmooth minimization. Math. Program. Study **17**, 77–90 (1982)
26. Mistakidis, E.S., Stavroulakis, G.E.: Nonconvex Optimization in Mechanics. Smooth and Nonsmooth Algorithms, Heuristics and Engineering Applications by the F.E.M. Kluwert Academic Publishers, Dordrecht (1998)
27. Moreau, J.J., Panagiotopoulos, P.D., Strang, G. (eds.): Topics in Nonsmooth Mechanics. Birkhäuser Verlag, Basel (1988)
28. Naniewicz, Z., Panagiotopoulos, P.D.: Mathematical Theory of Hemivariational Inequalities and Applications. Marcel Dekker, New York (1995)
29. Nocedal, J.: Updating quasi-Newton matrices with limited storage. Math. Comput. **35**(151), 773–782 (1980)
30. Nowak, M.P., Nrnberg, R., Römisch, W., Schultz, R., Westphalen, M.: Stochastic programming for power production and trading under uncertainty. In: W. Jger, H.J. Krebs (eds.) Mathematics — Key Technology for the Future, pp. 623–636. Springer, Berlin (2003)
31. Outrata, J., Kočvara, M., Zowe, J.: Nonsmooth Approach to Optimization Problems With Equilibrium Constraints. Theory, Applications and Numerical Results. Kluwert Academic Publisher, Dordrecht (1998)
32. Panagiotopoulos, P.D.: Hemivariational Inequalities. Springer-Verlag, New York (1993)
33. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**(1), 121–152 (1992)

34. Vlček, J., Lukšan, L.: Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. J. Optimization Theory Appl. **111**(2), 407–430 (2001)