Limited Memory Bundle Method for Large-Scale Nonsmooth Optimization: Convergence Analysis

M. Haarala K. Miettinen M. M. Mäkelä

Department of Mathematical Information Technology,
University of Jyväskylä, P.O. Box 35 (Agora),
FIN-40014 University of Jyväskylä, Finland.

In this paper we describe a limited memory bundle method for solving large nonsmooth (nondifferentiable) optimization problems. The method is a hybrid of the nonsmooth variable metric bundle method and the smooth limited memory variable metric method. We prove the global convergence of the method for a locally Lipschitz continuous objective function.

Keywords: Nondifferentiable programming, large-scale optimization, bundle methods, variable metric methods, limited memory methods, global convergence.

1 Introduction

In this paper, we describe a limited memory bundle algorithm for solving large nonsmooth (nondifferentiable) unconstrained optimization problems. We write this problem as

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

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) arise in many fields of applications, for example, in image restoration (see, e.g., [8]) and in optimal control (see, e.g., [12]). The direct application of smooth gradient-based methods to nonsmooth problems may lead to a failure in convergence, in optimality conditions, or in gradient approximation (see [9]). On the other hand, direct methods, for example, Powel's method (see, e.g., [3]) employing no derivative

information, are quite unreliable and become inefficient when the size of the problem increases. Thus, we need special tools for solving nonsmooth optimization problems.

At the moment, various versions of bundle methods (see, e.g., [6, 7, 11, 12]) are regarded as the most effective and reliable globally convergent methods for nonsmooth optimization. They are based on the assumption that at every point $\mathbf{x} \in \mathbb{R}^n$, we can evaluate the value of the objective function $f(\mathbf{x})$ and an arbitrary subgradient $\boldsymbol{\xi} \in \mathbb{R}^n$ from the subdifferential (see [2])

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

where "conv" denotes the convex hull of a set.

The basic idea of bundle methods is to approximate the subdifferential of the objective function by gathering the subgradients from previous iterations into a bundle. A search direction for the objective function can be determined by solving a quadratic direction finding problem (see, e.g., [12]) and the global convergence of bundle methods with a limited number of stored subgradients can be guaranteed by using a subgradient aggregation strategy [7], which accumulates information from the previous iterations.

In their present form, bundle methods are efficient for small- and medium-scale problems. However, their computational demand expands in large-scale problems with more than 500 variables (see [4]). This is explained by the fact that bundle methods need relatively large bundles to be capable of solving the problems efficiently. In other words, the size of the bundle has to be approximately the same as the number of variables and, thus, the quadratic direction finding problem becomes very time-consuming to solve.

In variable metric bundle methods introduced by Lukšan and Vlček [10, 14], the search direction is calculated by using the variable metric approximation of the inverse of the Hessian matrix. Thus, the quite complicated quadratic direction finding problem appearing in standard bundle methods needs not to be solved. Furthermore, the subgradient aggregation is done by using only three subgradients and, thus, the size of the bundle need not to grow with the dimension of the problem. However, variable metric bundle methods use dense approximations of the Hessian matrix to calculate the search direction and, thus, due to matrix operations also these methods become inefficient when the dimension of the problem increases.

In [4] we have introduced a limited memory bundle method for large-scale nonsmooth optimization. The method proposed is a hybrid of the variable metric bundle methods [10, 14] and the limited memory variable metric methods

(see, e.g., [1, 13]), where the latter have been developed for smooth large-scale optimization. The new method uses all 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 the standard bundle methods need not to be solved and the number of stored subgradients need not to grow with the dimension of the problem. Furthermore, the method uses only few vectors to represent the variable metric approximation of the Hessian matrix and, thus, it avoids storing and manipulating large matrices as is the case in variable metric bundle methods (see [10, 14]). These improvements make the limited memory bundle method suitable for large-scale optimization. Namely, the number of operations needed for the calculation of the search direction and the aggregate values is only linearly dependent on the number of variables while, for example, with the original variable metric bundle method, this dependence is quadratic.

In order to prove the global convergence of the limited memory bundle method some modifications had to be made to the algorithm introduced in [4]. In this paper, we first present a modified limited memory bundle algorithm. Then, we propose a special line search procedure, which is slightly modified from that given in [14] and used in [4]. In addition, we give a modified limited memory SR1 update, which guarantees the conditions required for the global convergence in the consecutive null steps.

This paper is organized as follows: In the following section, we give the modified limited memory bundle algorithm together with the line search procedure and the limited memory matrix updating. In Section 3, we prove the global convergence of the method and, finally, in Section 4, we conclude and give some ideas of the further development.

2 Limited Memory Bundle Method

In this section we first introduce the modified limited memory bundle algorithm. After that we propose a special line search procedure which provides step sizes satisfying the conditions required for global convergence. We also give the algorithms for limited memory BFGS and SR1 updates, since these algorithms differ from those given in [4]. The changes in BFGS update have no effect to the global convergence of the method but the SR1 update is modified such that it guarantees the sequence of stopping criterion (w_k) to be nonincreasing in the consecutive null steps. For more detailed description of other

properties of limited memory bundle method we refer to [4].

The algorithm given below generates a sequence of basic points $(\mathbf{x}_k) \in \mathbb{R}^n$ that, in the convex case, converges to a global minimum of a objective function $f : \mathbb{R}^n \to \mathbb{R}$. In nonconvex case, the algorithm is only guaranteed to find a stationary point of the objective function (that is, a point $\mathbf{x} \in \mathbb{R}^n$ satisfying $\mathbf{0} \in \partial f(\mathbf{x})$). In addition to the basic points, the algorithm generates a sequence of auxiliary points $(\mathbf{y}_k) \in \mathbb{R}^n$. A new iteration point $\mathbf{x}_{k+1}$ and a new auxiliary point $\mathbf{y}_{k+1}$ are produced by using a special line search procedure such that

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

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

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

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 $\mathbf{x}_k$. If the required condition (3) is satisfied, we set $\mathbf{x}_{k+1} = \mathbf{y}_{k+1}$ and a serious step is taken. Otherwise, a null step is taken if

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

where $\varepsilon_R^k \in (\varepsilon_L^k, 1/2)$ is a line search parameter, $\boldsymbol{\xi}_{k+1} \in \partial f(\mathbf{y}_{k+1})$, and β_{k+1} is the subgradient locality measure similar to bundle methods (see, e.g., [11]). In the case of a null step, we set $\mathbf{x}_{k+1} = \mathbf{x}_k$ but information about the objective function is increased because of the auxiliary point $\mathbf{y}_{k+1}$ and the auxiliary subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\mathbf{y}_{k+1})$ that we store.

The aggregation strategy used with the limited memory bundle method is similar to that with the original variable metric bundle methods [10, 14]. We denote by m the lowest index j satisfying $\mathbf{x}_j = \mathbf{x}_k$ (that is, m is the index of the iteration after the latest serious step). Suppose that we have the current subgradient $\boldsymbol{\xi}_m \in \partial f(\mathbf{x}_k)$, the auxiliary subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\mathbf{y}_{k+1})$, 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 of these three subgradients,

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

where the multipliers $\lambda_i^k \geq 0$ for $i \in \{1,2,3\}$ and $\sum_{i=1}^3 \lambda_i = 1$ 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 quite complicated quadratic direction finding problem appearing in standard bundle methods (see, e.g., [12]). Note that the aggregate values are computed only if the last step was a null step. Otherwise, we set $\tilde{\boldsymbol{\xi}}_{k+1} = \boldsymbol{\xi}_{k+1} \in \partial f(\mathbf{x}_{k+1})$.

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

Algorithm 1. (Limited Memory Bundle Method).

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

Step 0: (*Initialization.*) Choose a starting point $\mathbf{x}_1 \in \mathbb{R}^n$ and set an initial matrix $D_1 = I$. Set $\beta_1 = 0$ and $\mathbf{y}_1 = \mathbf{x}_1$. Compute

$$f_1 = f(\mathbf{x}_1) \qquad \text{and}$$
$$\boldsymbol{\xi}_1 \in \partial f(\mathbf{x}_1).$$

Set the correction indicator $i_C = 0$. Set 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 $i_{CN} = 0$ and an index for the serious step $m = k$.

Step 2: (*Direction finding.*) Compute

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

by the limited memory BFGS update if $m = k$ (Algorithm 3) and by the limited memory SR1 update, otherwise (Algorithm 4).

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

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

(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

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

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

If $w_k < \varepsilon$ and $q_k < \varepsilon$, then stop.

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

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

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

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

Set $\mathbf{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ and $\mathbf{s}_k = \mathbf{y}_{k+1} - \mathbf{x}_k = t_R^k \theta_k \mathbf{d}_k$. 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, calculate the locality measure

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

Step 6: (*Aggregation.*) Determine multipliers $\lambda_i^k \geq 0$, $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^k = 1$ 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 $D_k = D_k + \varrho I$ if $i_C = 1$, and D_k is calculated by the same updating formula as in Step 2. 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 direction vector length (see Step 5 in Algorithm 1) and the matrices $B_i = D_i^{-1}$ (see Step 3 in Algorithm 1) are required. 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 search direction $\mathbf{d}_k$ and the aggregate values $\tilde{\boldsymbol{\xi}}_{k+1}$ and $\tilde{\beta}_{k+1}$ are calculated using the limited memory approach.

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 corresponding function values and subgradients. The procedure used is exactly the same as in the original variable metric bundle method for nonconvex objective functions (see [14]). Since the aggregation procedure (see Step 6 in Algorithm 1) uses only three subgradients and two locality measures to calculate the new aggregate values, the minimum size of the bundle is two and a larger bundle (if it is employed) is used only for the selection of the initial step size.

Next we present a line search algorithm, which is used to determine the step sizes t_L^k and t_R^k in limited memory bundle methods. The line search procedure used is quite similar to that in the original variable metric bundle method [14]. However, in order to guarantee the global convergence of the method we use scaled line search parameters ε_R^k, ε_L^k, ε_A^k, and ε_T^k instead of fixed ones (see, e.g., [14]). Furthermore, in order to avoid many consecutive null steps, we have added an additional interpolation step (Step 3 in Algorithm 2). That is, 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, we rather try to find a step size suitable for a serious step (that is, (3) is valid) even if condition (4) required for a null step was satisfied. This additional interpolation step has no influence to the convergence properties but it has significant affect to the efficiency of the method. Neither the choice of the interpolation procedure (see Step 5 in Algorithm 2) has effect to the convergence properties. Similarly to original variable metric bundle method (see [14]) we combine quadratic interpolation with the bisection.

Algorithm 2. (Line Search).

Data: Suppose that we have the current iteration point $\mathbf{x}_k$, the current search direction $\mathbf{d}_k$, the current scaling parameter $\theta_k \in (0, 1]$ and positive initial line search parameters $\varepsilon_R^I \in (0, 1/2)$, $\varepsilon_L^I \in (0, \varepsilon_R^I)$, $\varepsilon_A^I \in (0, \varepsilon_R^I - \varepsilon_L^I)$, 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 $\gamma \geq 0$ ($\gamma = 0$ if f is convex), the locality measure parameter $\omega \geq 1$, the desirable amount of descent w_k, an

interpolation parameter $\kappa \in (0, 1/2)$, and the maximum number for additional interpolations int_{max}. 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

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

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

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

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

Step 2: (*Serious Step.*) If

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

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 terminate the computation.

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

Step 4: (*Null Step.*) If

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

then set $t_R^k = t$, $t_L^k = 0$ and terminate the computation.

Step 5: (*Interpolation.*) Choose

$$t \in [t_A + \kappa(t_U - t_A), t_U - \kappa(t_U - t_A)]$$

by using some interpolation procedure and go to Step 1.

It can be proved under some semi-smoothness hypotheses that Algorithm 2 terminates in a finite number of iterations (the proof is similar to that given in [14]). 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

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

and, in case of $t_L^k = 0$ (null step), also condition (4).

Finally, we need to consider how to update the approximation D_k of the inverse of the Hessian matrix and, thus, how to find the search direction $\mathbf{d}_k$. The basic idea of the limited memory matrix updating is that instead of storing the matrices D_k, we use the information of the last few iterations to implicitly define the approximation of the inverse of the Hessian. This is done by storing a certain number of correction pairs $(\mathbf{s}_i, \mathbf{u}_i)$, $(i < k)$, obtained in Step 5 of Algorithm 1. When the storage space available is used up, the oldest corrections are deleted to make room for new ones.

Let us denote by m_c the maximum number of stored corrections supplied by user $(3 \leq m_c)$ and by $m_k = \min\{k-1, m_c\}$ the current number of stored corrections. The $n \times m_k$ -dimensional correction matrices S_k and U_k are defined by

$$S_k = \begin{bmatrix} \mathbf{s}_{k-m_k} & \dots & \mathbf{s}_{k-1} \end{bmatrix} \qquad \text{and}$$
$$U_k = \begin{bmatrix} \mathbf{u}_{k-m_k} & \dots & \mathbf{u}_{k-1} \end{bmatrix}.$$

These correction matrices are used to implicitly define the approximation of the inverse of the Hessian matrix at each iteration. When a new auxiliary point $\mathbf{y}_{k+1}$ is generated, the new correction matrices S_{k+1} and U_{k+1} are obtained by deleting the oldest corrections $\mathbf{s}_{k-m_k}$ and $\mathbf{u}_{k-m_k}$ from S_k and U_k if $m_{k+1} = m_k$ (that is, $k > m_c$) and by adding the most recent corrections $\mathbf{s}_k$ and $\mathbf{u}_k$ to the matrices. Thus, except for the first few iterations, we always have the m_c most recent correction pairs $(\mathbf{s}_i, \mathbf{u}_i)$ available.

We define the inverse limited memory BFGS update by the formula (see [1])

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

Here, R_k is an upper triangular matrix of order m_k given by the form

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

C_k is a diagonal matrix of order m_k such that

$$C_k = \operatorname{diag}[\mathbf{s}_{k-m_k}^T \mathbf{u}_{k-m_k}, \dots, \mathbf{s}_{k-1}^T \mathbf{u}_{k-1}],$$

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

$$\vartheta_k = \frac{\mathbf{u}_{k-1}^T \mathbf{s}_{k-1}}{\mathbf{u}_{k-1}^T \mathbf{u}_{k-1}}. \tag{14}$$

In addition, we define the inverse limited memory SR1 update (see [1]) by

$$D_k = \vartheta_k I - (\vartheta_k U_k - S_k)(\vartheta_k U_k^T U_k - R_k - R_k^T + C_k)^{-1}(\vartheta_k U_k - S_k)^T, \quad (15)$$

where instead of (14) we use the value $\vartheta_k = 1$ for every k.

Next, we describe some procedures for updating the limited memory BFGS and SR1 matrices. In addition to the two $n \times m_k$ -matrices S_k and U_k, we store the $m_k \times m_k$ -matrices R_k, $U_k^T U_k$, and C_k and the m_{k-1}-vectors $S_{k-1}^T \boldsymbol{\xi}_m$ and $U_{k-1}^T \boldsymbol{\xi}_m$ from the previous iteration. Since in practice m_k is clearly smaller than n, the storage space required by these three auxiliary matrices and two vectors is insignificant but the savings in computational efforts are considerable. A detailed description of updating these matrices and vectors can be found in [4].

We first give an efficient algorithm for updating the limited memory BFGS matrix D_k and for computing the search direction $\mathbf{d}_k = -D_k \boldsymbol{\xi}_k$. This algorithm is used whenever the previous step was a serious step. Note that after a serious step the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k \in \partial f(\mathbf{x}_k)$ and, thus, the correction vectors obtained at the previous iteration in Step 5 of Algorithm 1 can be equally expressed as $\mathbf{s}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1}$ and $\mathbf{u}_{k-1} = \boldsymbol{\xi}_k - \boldsymbol{\xi}_{k-1}$.

Algorithm 3. (BFGS Updating and Direction Finding).

Data: Suppose that the number of current corrections is m_{k-1} and the maximum number of stored corrections is m_c. Suppose that we have the most recent corrections $\mathbf{s}_{k-1}$ and $\mathbf{u}_{k-1}$ (from previous iteration), the current subgradient $\boldsymbol{\xi}_k \in \partial f(\mathbf{x}_k)$, the previous subgradient $\boldsymbol{\xi}_{k-1} \in \partial f(\mathbf{x}_{k-1})$, the $n \times m_{k-1}$ -matrices S_{k-1} and U_{k-1}, the $m_{k-1} \times m_{k-1}$ -matrices R_{k-1}, $U_{k-1}^T U_{k-1}$, and C_{k-1}, the previous scaling parameter ϑ_{k-1}, and the m_{k-1}-vectors $S_{k-1}^T \boldsymbol{\xi}_{k-1}$ and $U_{k-1}^T \boldsymbol{\xi}_{k-1}$ available.

Step 1: (*Positive Definiteness*) If

$$\mathbf{u}_{k-1}^T \mathbf{s}_{k-1} > 0, \quad (16)$$

then set $m_k = \min\{ m_{k-1} + 1, m_c \}$ and update the matrices (i.e., go to Step 2). Otherwise, skip the updates, that is, set $S_k = S_{k-1}$, $U_k = U_{k-1}$, $R_k = R_{k-1}$, $U_k^T U_k = U_{k-1}^T U_{k-1}$, $C_k = C_{k-1}$, $\vartheta_k = \vartheta_{k-1}$, and $m_k = m_{k-1}$, compute $S_k^T \boldsymbol{\xi}_k$ and $U_k^T \boldsymbol{\xi}_k$, and go to Step 7.

Step 2: Obtain S_k and U_k by updating S_{k-1} and U_{k-1}.

Step 3: Compute and store m_k-vectors $S_k^T \boldsymbol{\xi}_k$ and $U_k^T \boldsymbol{\xi}_k$.

Step 4: Compute m_k-vectors $S_k^T \mathbf{u}_{k-1}$ and $U_k^T \mathbf{u}_{k-1}$ by using the fact

$$\mathbf{u}_{k-1} = \boldsymbol{\xi}_k - \boldsymbol{\xi}_{k-1}.$$

Step 5: Update $m_k \times m_k$ -matrices R_k, $U_k^T U_k$, and C_k.

Step 6: If $\mathbf{u}_{k-1}^T \mathbf{u}_{k-1} > 0$, compute ϑ_k

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

Note that both $\mathbf{u}_{k-1}^T \mathbf{s}_{k-1}$ and $\mathbf{u}_{k-1}^T \mathbf{u}_{k-1}$ have already been calculated. Otherwise, set $\vartheta_k = 1.0$.

Step 7: (*Intermediate Values*) Solve two intermediate values $\mathbf{p}_1 \in \mathbb{R}^{m_k}$ and $\mathbf{p}_2 \in \mathbb{R}^{m_k}$ from the linear equations

$$\begin{aligned} R_k \mathbf{p}_1 &= S_k^T \boldsymbol{\xi}_k, \\ R_k^T \mathbf{p}_2 &= C_k \mathbf{p}_1 + \vartheta_k U_k^T U_k \mathbf{p}_1 - \vartheta_k U_k^T \boldsymbol{\xi}_k. \end{aligned}$$

Step 8: (*Search Direction*) Compute

$$\mathbf{d}_k = \vartheta_k U_k \mathbf{p}_1 - S_k \mathbf{p}_2 - \vartheta_k \boldsymbol{\xi}_k.$$

Note that condition (16) assures the positive definiteness of the matrices obtained by the limited memory BFGS update (see, e.g., [1]).

If the previous step was a null step, then $\tilde{\boldsymbol{\xi}}_k \neq \boldsymbol{\xi}_k$ and, thus, there is no use to utilize the difference $\mathbf{u}_{k-1} = \boldsymbol{\xi}_k - \boldsymbol{\xi}_m$ in the calculations of $S_k^T \mathbf{u}_{k-1}$ and $U_k^T \mathbf{u}_{k-1}$ (see Step 4 in Algorithm 3). Furthermore, in order to guarantee the global convergence of the method, the sequence (w_k) has to be nonincreasing in consecutive null steps (that is, $w_k \leq w_{k-1}$). Thus, the condition

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

has to be satisfied at each time there occurs more than one consecutive null step. In practice, these two means that there are some more calculations required in the case of null steps than in the case of serious steps.

We now give an efficient algorithm for updating the limited memory SR1 matrix D_k and for computing the search direction $\mathbf{d}_k = -D_k \tilde{\boldsymbol{\xi}}_k$. Along with Step 3 of Algorithm 1, this procedure guarantees that condition (17) is valid even when the correction ϱI is added to the new matrix D_k (see Lemma 10). This algorithm is used whenever the previous step was a null step.

Algorithm 4. (SR1 Updating and Direction Finding).

Data: Suppose that the number of current corrections is m_{k-1} and the maximum number of stored corrections is m_c. Suppose that we have the most recent corrections $\mathbf{s}_{k-1}$ and $\mathbf{u}_{k-1}$ (from previous iteration), the current aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$, the previous aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k-1}$, the previous search direction $\mathbf{d}_{k-1}$, the $n \times m_{k-1}$ -matrices S_{k-1} and U_{k-1}, the $m_{k-1} \times m_{k-1}$ -matrices R_{k-1}, $U_{k-1}^T U_{k-1}$, and C_{k-1}, and the previous scaling parameter ϑ_{k-1} available. In addition, suppose that we have the number of consecutive null steps $i_{null} = k - m \geq 1$.

Step 1: (*Initial Vectors and Initialization*) Compute m_{k-1}-vectors $S_{k-1}^T \tilde{\boldsymbol{\xi}}_k$ and $U_{k-1}^T \tilde{\boldsymbol{\xi}}_k$. Set $\vartheta_k = 1.0$ and $i_{up} = 0$.

Step 2: (*Positive Definiteness*) If

$$-\mathbf{d}_{k-1}^T \mathbf{u}_{k-1} - \tilde{\boldsymbol{\xi}}_{k-1}^T \mathbf{s}_{k-1} \geq 0,$$

then skip the updates, that is, set $S_k = S_{k-1}$, $U_k = U_{k-1}$, $R_k = R_{k-1}$, $U_k^T U_k = U_{k-1}^T U_{k-1}$, $C_k = C_{k-1}$, $S_k^T \tilde{\boldsymbol{\xi}}_k = S_{k-1}^T \tilde{\boldsymbol{\xi}}_k$, $U_k^T \tilde{\boldsymbol{\xi}}_k = U_{k-1}^T \tilde{\boldsymbol{\xi}}_k$, and $m_k = m_{k-1}$ and go to Step 8.

Otherwise, set $m_k = \min\{ m_{k-1} + 1, m_c \}$ and calculate $\mathbf{s}_{k-1}^T \tilde{\boldsymbol{\xi}}_k$ and $\mathbf{u}_{k-1}^T \tilde{\boldsymbol{\xi}}_k$.

Step 3: (*Update Conditions*) If either

$$\begin{aligned} &i_{null} = 1, \qquad \text{or} \\ &m_k < m_c, \end{aligned}$$

then update the matrices, that is, go to Step 4. Otherwise, solve the linear equation

$$\begin{aligned} (\vartheta_{k-1} U_{k-1}^T U_{k-1} - R_{k-1} - R_{k-1}^T + C_{k-1})\mathbf{p} \\ = \vartheta_{k-1} U_{k-1}^T \tilde{\boldsymbol{\xi}}_k - S_{k-1}^T \tilde{\boldsymbol{\xi}}_k. \end{aligned}$$

to obtain $\mathbf{p} \in \mathbb{R}^{m_{k-1}}$. Calculate the vector $\mathbf{z} \in \mathbb{R}^n$ from

$$\mathbf{z} = \vartheta_{k-1} \tilde{\boldsymbol{\xi}}_k - (\vartheta_{k-1} U_{k-1} - S_{k-1})\mathbf{p},$$

and the scalar

$$a = \tilde{\boldsymbol{\xi}}_k^T \mathbf{z}.$$

Set $i_{up} = 1$.

Step 4: Obtain S_k and U_k by updating S_{k-1} and U_{k-1}.

Step 5: Compute m_k-vectors $S_k^T \mathbf{u}_{k-1}$ and $U_k^T \mathbf{u}_{k-1}$.

Step 6: Update $m_k \times m_k$ -matrices R_k, $U_k^T U_k$, and C_k.

Step 7: Construct m_k-vectors $S_k^T \tilde{\boldsymbol{\xi}}_k$ and $U_k^T \tilde{\boldsymbol{\xi}}_k$ using $S_{k-1}^T \tilde{\boldsymbol{\xi}}_k$, $U_{k-1}^T \tilde{\boldsymbol{\xi}}_k$, $\mathbf{s}_{k-1}^T \tilde{\boldsymbol{\xi}}_k$, and $\mathbf{u}_{k-1}^T \tilde{\boldsymbol{\xi}}_k$.

Step 8: (*Intermediate Value*) Solve $\mathbf{p} \in \mathbb{R}^{m_k}$ from the linear equation

$$(\vartheta_k U_k^T U_k - R_k - R_k^T + C_k)\mathbf{p} = \vartheta_k U_k^T \tilde{\boldsymbol{\xi}}_k - S_k^T \tilde{\boldsymbol{\xi}}_k.$$

Step 9: (*Search Direction*) Compute

$$\mathbf{d}_k = -\vartheta_k \tilde{\boldsymbol{\xi}}_k + (\vartheta_k U_k - S_k)\mathbf{p}.$$

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

$$b = \tilde{\boldsymbol{\xi}}_k \mathbf{d}_k,$$

and in case of

$$b + a < 0,$$

set $m_k = m_{k-1}$, $S_k = S_{k-1}$, $U_k = U_{k-1}$, $R_k = R_{k-1}$, $U_k^T U_k = U_{k-1}^T U_{k-1}$, $C_k = C_{k-1}$, and

$$\mathbf{d}_k = -\mathbf{z}.$$

LEMMA 1. *The condition (see Algorithm 4, Step 2)*

$$-\mathbf{d}_i^T \mathbf{u}_i - \tilde{\boldsymbol{\xi}}_i^T \mathbf{s}_i < 0 \qquad \textit{for all } i = 1, \ldots, k-1 \tag{18}$$

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 (18) 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., [3]) to (15) we obtain

$$B_k = B_1 + (U_k - B_1 S_k)(L_k + L_k^T + C_k - S_k^T B_1 S_k)^{-1}(U_k - B_1 S_k)^T, \tag{19}$$

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

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

We denote

$$Q_k = \begin{bmatrix} \mathbf{q}_{k-m_k} & \dots & \mathbf{q}_{k-1} \end{bmatrix} = U_k - B_1 S_k$$

and

$$N_k = L_k + L_k^T + C_k - S_k^T B_1 S_k.$$

Assume that B_{k-1} is positive definite for some $k > 1$ and that condition (18) is valid for $i = 1, \dots, k-1$. We prove that also the new 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 (19) can be rewritten in the form

$$\begin{aligned} B_+ &= B_1 + Q_+ N_+^{-1} Q_+^T \\ &= B' + \frac{(\mathbf{u} - B'\mathbf{s})\,(\mathbf{u} - B'\mathbf{s})^T}{\mathbf{s}^T\,(\mathbf{u} - B'\mathbf{s})}, \end{aligned} \tag{20}$$

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

It can be easily seen from (20) that the new matrix B_+ is positively definite if B' is positively definite and the denominator $\mathbf{s}^T\,(\mathbf{u} - B'\mathbf{s}) > 0$.

We first prove that B' is positively definite. The current matrix B is positive definite by assumption and, thus, due to condition (18), $QN^{-1}Q^T$, N^{-1}, and N are also positive definite. The positive definiteness of N implies the positive definiteness of N' as a minor 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.

Condition (18) implies that for all $i = 1, \dots, k-1$ we have

$$\mathbf{d}_i^T \mathbf{u}_i > t_R^i \theta_i \tilde{\boldsymbol{\xi}}_i^T D_i \tilde{\boldsymbol{\xi}}_i = t_R^i \theta_i \tilde{\boldsymbol{\xi}}_i^T D_i B_i D_i \tilde{\boldsymbol{\xi}}_i = t_R^i \theta_i \mathbf{d}_i^T B_i \mathbf{d}_i, \tag{21}$$

where $t_R^i > 0$ is the step size and $\theta_i \in (0, 1]$ is the scaling parameter for the direction vector $\mathbf{d}_i$. From (21) and the fact that $\mathbf{s} = t_R \theta \mathbf{d}$ we obtain

$$\mathbf{s}^T(\mathbf{u} - B\mathbf{s}) > 0.$$

Now, for $k \leq m_c + 1$ the current positive definite matrix B is equal to B' and for $k > m_c + 1$ it can be given in a form

$$\begin{aligned} B &= B_1 + QN^{-1}Q^T \\ &= B_1 + \begin{bmatrix} \mathbf{q}_- & Q' \end{bmatrix} \begin{bmatrix} \mathbf{s}_-^T\mathbf{q}_- & \mathbf{q}_-^T S' \\ S'^T\mathbf{q}_- & N' \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{q}_-^T \\ Q'^T \end{bmatrix} \\ &= B_1 + \begin{bmatrix} \mathbf{q}_- & Q' \end{bmatrix} \begin{bmatrix} 1/\delta & -\mathbf{q}_-^T S' N'^{-1}/\delta \\ -N'^{-1}S'^T\mathbf{q}_-/\delta & N'^{-1} + N'^{-1}S'^T\mathbf{q}_-\mathbf{q}_-^T S' N'^{-1}/\delta \end{bmatrix} \begin{bmatrix} \mathbf{q}_-^T \\ Q'^T \end{bmatrix} \\ &= B' + (\mathbf{q}_- - Q'N'^{-1}S'^T\mathbf{q}_-)(\mathbf{q}_- - Q'N'^{-1}S'^T\mathbf{q}_-)^T/\delta, \end{aligned}$$

where the subscript "$-$" denotes the index $k - m_k - 1$, matrix S' is formed by deleting the first column from S, and the denominator

$$\delta = \mathbf{s}_-^T\mathbf{q}_- - \mathbf{q}_-^T S' N'^{-1} S'^T \mathbf{q}_-$$

is greater than zero, since matrix N^{-1} is positive definite (that is, the upper left term $1/\delta$ has to be greater than zero). Thus, in all cases, we have $\mathbf{x}^T(B - B')\mathbf{x} \geq 0$ for all $\mathbf{x} \neq 0$, and the following inequality holds for the denominator of formula (20):

$$\mathbf{s}^T(\mathbf{u} - B'\mathbf{s}) \geq \mathbf{s}^T(\mathbf{u} - B\mathbf{s}) > 0.$$

Therefore, the new matrix B_+ and its inverse D_+ are positive definite. □

In order to use both Algorithms 3 and 4 with the same stored information, some modifications have to be made. Firstly, during the limited memory SR1 update we have to update the m_k-vectors $S_k^T\boldsymbol{\xi}_m$ and $U_k^T\boldsymbol{\xi}_m$ (that is, $S_{k-1}^T\boldsymbol{\xi}_{k-1}$ and $U_{k-1}^T\boldsymbol{\xi}_{k-1}$ to the next limited memory BFGS update). Furthermore, since we use the same correction matrices S_k and U_k for the calculations of both the BFGS and the SR1 updates, both the positive definiteness conditions (16) and (18) have to be valid in each case before we update the matrices. Nevertheless, it can be easily seen from (21) provided by positive definiteness of B_i that condition (18) implies (16) and, thus, we only have to check the validity of (18). However, numerical experiments have showed that the simple skipping of the BFGS update (see Algorithm 3, Step 1) if condition (18) required for the SR1 update is not satisfied, makes the method quite inefficient. Therefore, in case of BFGS update, the new search direction $\mathbf{d}_k$ is calculated conventionally using the most recent corrections $\mathbf{s}_{k-1}$ and $\mathbf{u}_{k-1}$ whenever the required positive definiteness condition (16) is valid but the matrices are not updated, that is, the correction vectors are not stored, unless both the conditions (16) and (18) are satisfied. In practice, this means that the correction matrices S_k and U_k may actually include some indices smaller than $k - m_k$ due to skipping of updates and that, in the case of the BFGS update, the number of the current corrections used may be $m_k = m_c + 1$.

The new limited memory bundle method uses a limited memory approach to calculate the search direction and it requires only three subgradients and two locality measures to calculate the new aggregate values. Thus, the time-consuming quadratic direction finding problem appearing in standard bundle methods (see, e.g., [12]) needs not to be solved and the size of the bundle needs not to increase with the dimension of the problem. Furthermore, both the search direction $\mathbf{d}_k$ and the aggregate values $\tilde{\boldsymbol{\xi}}_{k+1}$ and $\tilde{\beta}_{k+1}$ can be computed implicitly using at most $O(nm_c)$ operations. Assuming $m_c \ll n$, this is much less than $O(n^2)$ operations needed with the original variable metric bundle method, which stores and manipulates the whole matrix D_k. These improvements make the limited memory bundle method suitable for large-scale problems. This assertion is supported by numerical experiments presented in [4].

3 Convergence Analysis

In this section, we prove the global convergence of Algorithm 1. We assume that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous and that the level set $\{ \mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \leq f(\mathbf{x}_1) \}$ is bounded. In addition, we assume that each execution of the line search procedure is finite.

We start by giving a necessary condition for a locally Lipschitz continuous objective function to attain its local minimum in an unconstrained case. For a convex function this condition is also sufficient and the minimum is global.

THEOREM 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz continuous at $\mathbf{x} \in \mathbb{R}^n$. If f attains its local minimum at $\mathbf{x}$, then $\mathbf{x}$ is a stationary point, that is,*

$$\mathbf{0} \in \partial f(\mathbf{x}).$$

PROOF. See, e.g., [12].

Now, since 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 $(\mathbf{x}_k)$ for which accumulation points are stationary for f. Naturally, we assume that the final accuracy tolerance $\varepsilon = 0$.

We start the convergence analysis by giving three technical results (Lemmas 3, 4, and 5). After that, we prove (Theorem 6) that $w_k = 0$ and $q_k = 0$ imply that the point $\mathbf{x}_k$ is a stationary point for the objective function. For infinite sequence $(\mathbf{x}_k)$, we first show (Lemma 7) that the conditions $(q_k) \to 0$ and $(w_k) \to 0$ are equivalent due to correction (6) and, thus, we can restrict

the consideration to the stopping parameter w_k. Then we show (Lemma 8) that if $(\mathbf{x}_k)_{k\in\mathcal{K}} \to \bar{\mathbf{x}}$ and $(w_k)_{k\in\mathcal{K}} \to 0$ for some subset $\mathcal{K} \subset \{1, 2, \ldots\}$, then the accumulation point $\bar{\mathbf{x}}$ is 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 technical Lemma 9 and the fact that the sequence (w_k) is nonincreasing in the consecutive null steps due to additional testing procedure during the limited memory SR1 update (see Lemma 10), we prove that the indefinite sequence of consecutive null steps with $\mathbf{x}_k = \mathbf{x}_m$ implies $\mathbf{0} \in \partial f(\mathbf{x}_m)$ (Lemma 11). Finally, in Theorem 12 we combine all the results obtained and show that every accumulation point of $(\mathbf{x}_k)$ is stationary for the objective function.

The convergence analysis of limited memory bundle method is very similar to that of original variable metric bundle method for nonconvex objective functions (see [14]). In fact, Lemmas 4, 5, and 9 and their proofs are exactly the same as those given in [14] (the proof of Lemma 9 can be found from [10]) and also Lemmas 3, 8, 11, and Theorems 6, and 12 are very similar to the corresponding of original variable metric bundle method. However, we give those Lemmas and Theorems (with their proofs) here to make the convergence analysis of limited memory bundle method self-contained.

There are two main differences between the convergence analysis of original variable metric bundle method and limited memory bundle method. The first one is that in limited memory bundle method we have two different stopping parameters w_k and q_k (see Step 4 in Algorithm 1) instead of only one (see [14]). However, Lemma 7 shows 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 of original variable metric bundle method. Furthermore, in limited memory bundle method 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 4). Lemma 10 shows that along with Step 3 of Algorithm 1, the procedure used in Algorithm 4 guarantees that condition (17) is valid even when the correction ϱI is added to new matrix D_k.

LEMMA 3. *At the kth iteration of Algorithm 1, we have*

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

and

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

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

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

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

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

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

By (9) and since $\mathbf{x}_{k+1} = \mathbf{x}_k$ for null steps, and since $\beta_{k+1} = 0$, and $\|\mathbf{y}_{k+1} - \mathbf{x}_{k+1}\| = 0$ for serious steps, we always have

$$\beta_{k+1} \geq \gamma\|\mathbf{y}_{k+1} - \mathbf{x}_{k+1}\|^\omega$$

for some $\gamma \geq 0$ and $\omega \geq 1$.

Now, we prove that condition (18) implies $\mathbf{u}_k^T(D_k\mathbf{u}_k - \mathbf{s}_k) > 0$. If condition (18) is valid for $k = k+1$, then $\tilde{\boldsymbol{\xi}_k} \neq \mathbf{0}$ (otherwise, we would have $-\mathbf{d}_k^T\mathbf{u}_k - \tilde{\boldsymbol{\xi}}_k^T\mathbf{s}_k = 0$). Using the positiveness of $\mathbf{u}_k^T\mathbf{s}_k$ (provided by the positive definiteness of D_k, in (21)), Cauchy's inequality, and the fact that we have $\mathbf{s}_k = t_R^k\theta_k\mathbf{d}_k$, $t_R^k > 0$ and $\theta_k \in (0, 1]$, we obtain

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

Therefore, $\mathbf{u}_k^T\mathbf{s}_k < \mathbf{u}_k^T D_k\mathbf{u}_k$. □

LEMMA 4. *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*

$$(\tilde{\boldsymbol{\xi}}_k, \tilde{\sigma}_k) = \sum_{j=1}^{k} \lambda^{k,j}(\boldsymbol{\xi}_j, \|\mathbf{y}_j - \mathbf{x}_k\|), \qquad \sum_{j=1}^{k} \lambda^{k,j} = 1, \qquad \textit{and}$$

$$\tilde{\beta}_k \geq \gamma\tilde{\sigma}_k^\omega.$$

PROOF. Let m be an index of the iteration after the latest serious step defined at Step 1 of Algorithm 1 (that is, $\mathbf{x}_j = \mathbf{x}_m$ for all $j = m, \ldots, k$). First we prove that there exist numbers $\lambda^{k,j} \geq 0$ for $j = m, \ldots, k$, such that

$$(\tilde{\boldsymbol{\xi}}_k, \tilde{\beta}_k) = \sum_{j=m}^{k} \lambda^{k,j} (\boldsymbol{\xi}_j, \beta_j), \qquad \sum_{j=m}^{k} \lambda^{k,j} = 1. \tag{22}$$

We prove this via induction. Suppose $k = m$. Then we set $\lambda^{m,m} = 1$, since $\tilde{\boldsymbol{\xi}}_m = \boldsymbol{\xi}_m$ and $\tilde{\beta}_m = 0$ at Step 1 of Algorithm 1 and we have set $\beta_m = 0$ at Step 5 at the previous iteration ($\beta_1 = 0$ due to initialization). Thus, the base case is valid. Now, suppose $k > m$, let $i \in \{m, \ldots, k-1\}$, and assume that (22) is valid for k replaced with i. We define

$$\begin{aligned} &\lambda^{i+1,m} = \lambda_1^i + \lambda_3^i \lambda^{i,m}, \\ &\lambda^{i+1,j} = \lambda_3^i \lambda^{i,j} \quad \text{for } j = m+1, \ldots, i, \qquad \text{and} \\ &\lambda^{i+1,i+1} = \lambda_2^i. \end{aligned}$$

Now, we have $\lambda^{i+1,j} \geq 0$ for all $j = m, \ldots, i+1$ and

$$\sum_{j=m}^{i+1} \lambda^{i+1,j} = \lambda_1^i + \lambda_3^i \left(\lambda^{i,m} + \sum_{j=m+1}^{i} \lambda^{i,j} \right) + \lambda_2^i = 1,$$

since $\sum_{j=m}^{i} \lambda^{i,j} = 1$ due to assumption and $\sum_{l=1}^{3} \lambda_l^i = 1$ (see Algorithm 1, Step 6).

Using relations (11) and (12), we obtain

$$\begin{aligned} (\tilde{\boldsymbol{\xi}}_{i+1}, \tilde{\beta}_{i+1}) &= \lambda_1^i (\boldsymbol{\xi}_m, 0) + \lambda_2^i (\boldsymbol{\xi}_{i+1}, \beta_{i+1}) + \sum_{j=m}^{i} \lambda_3^i \lambda^{i,j} (\boldsymbol{\xi}_j, \beta_j) \\ &= \sum_{j=m}^{i+1} \lambda^{i+1,j} (\boldsymbol{\xi}_j, \beta_j), \end{aligned}$$

due to $\beta_m = 0$ and, thus, condition (22) is valid for $i+1$.

Now, we define

$$\begin{aligned} &\lambda^{k,j} = 0 \quad \text{for } j = 1, \ldots, m-1 \qquad \text{and} \\ &\tilde{\sigma}_k = \sum_{j=1}^{k} \lambda^{k,j} \|\mathbf{y}_j - \mathbf{x}_k\|. \end{aligned}$$

Since $\mathbf{x}_j = \mathbf{x}_k$ for $j = m, \ldots, k$, we obtain

$$\tilde{\sigma}_k = \sum_{j=m}^{k} \lambda^{k,j} \|\mathbf{y}_j - \mathbf{x}_j\|,$$

and, thus, by (22), Lemma 3 and the convexity of the function $g \to \gamma g^\omega$ on $\mathbb{R}_+$ for $\gamma \geq 0$ and $\omega \geq 1$ we have

$$\begin{aligned}
\gamma \tilde{\sigma}_k^\omega &= \gamma \left(\sum_{j=m}^{k} \lambda^{k,j} \|\mathbf{y}_j - \mathbf{x}_j\| \right)^\omega \\
&\leq \sum_{j=m}^{k} \lambda^{k,j} \gamma \|\mathbf{y}_j - \mathbf{x}_j\|^\omega \\
&\leq \sum_{j=m}^{k} \lambda^{k,j} \beta_j \\
&= \tilde{\beta}_k.
\end{aligned}$$

□

LEMMA 5. *Let $\bar{\mathbf{x}} \in \mathbb{R}^n$ be given and suppose that there exist vectors $\bar{\mathbf{g}}$, $\bar{\boldsymbol{\xi}}_i$, $\bar{\mathbf{y}}_i$, and numbers $\bar{\lambda}_i \geq 0$ for $i = 1, \ldots, l$, $l \geq 1$, such that*

$$\begin{aligned}
&(\bar{\mathbf{g}}, 0) = \sum_{i=1}^{l} \bar{\lambda}_i (\bar{\boldsymbol{\xi}}_i, \|\bar{\mathbf{y}}_i - \bar{\mathbf{x}}\|), \\
&\bar{\boldsymbol{\xi}}_i \in \partial f(\bar{\mathbf{y}}_i), \quad i = 1, \ldots, l, \qquad \textit{and} \\
&\sum_{i=1}^{l} \bar{\lambda}_i = 1.
\end{aligned} \tag{23}$$

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

PROOF. Let $\mathcal{I} = \{\, i \mid i = 1, \ldots, l,\ \bar{\lambda}_i > 0 \,\}$. By (23) we have

$$\begin{aligned}
&\bar{\mathbf{y}}_i = \bar{\mathbf{x}} && \text{and} \\
&\bar{\boldsymbol{\xi}}_i \in \partial f(\bar{\mathbf{x}}) && \text{for all } i \in \mathcal{I}.
\end{aligned}$$

Thus, we obtain

$$\begin{aligned}
&\bar{\mathbf{g}} = \sum_{i \in \mathcal{I}} \bar{\lambda}_i \bar{\boldsymbol{\xi}}_i, \\
&\bar{\lambda}_i > 0, \qquad \text{for } i \in \mathcal{I}, \qquad \text{and} \\
&\sum_{i \in \mathcal{I}} \bar{\lambda}_i = 1,
\end{aligned}$$

and $\bar{\mathbf{g}} \in \partial f(\bar{\mathbf{x}})$ by the convexity of $\partial f(\bar{\mathbf{x}})$ (see [7]). □

THEOREM 6. *If Algorithm 1 terminates at the kth iteration, then the point* $\mathbf{x}_k$ *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 3 and Lemma 4. Now, by Lemma 4 and by using Lemma 5 with

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

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

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

LEMMA 7. *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 3 and, thus, it is obvious that $(w_k) \to 0$.

On the other hand, by Lemma 3 we have $w_k \ge 2\tilde{\beta}_k \ge 0$ and $w_k \ge \varrho \|\tilde{\boldsymbol{\xi}}_k\|^2$ 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 7 we can, from now on, restrict the consideration to the stopping parameter w_k.

LEMMA 8. *Suppose that the level set* $\{\, \mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \le f(\mathbf{x}_1) \,\}$ *is bounded. Then, the sequences* $(\mathbf{y}_k)$ *and* $(\boldsymbol{\xi}_k)$ *are also bounded. If, in addition, there exist a point* $\bar{\mathbf{x}} \in \mathbb{R}^n$ *and an infinite set* $\mathcal{K} \subset \{1, 2, \ldots\}$ *such that* $(\mathbf{x}_k)_{k \in \mathcal{K}} \to \bar{\mathbf{x}}$ *and* $(w_k)_{k \in \mathcal{K}} \to 0$, *then* $\mathbf{0} \in f(\bar{\mathbf{x}})$.

PROOF. The sequence $(\mathbf{x}_k)$ is bounded by assumption and the monotonicity of the sequence (f_k) obtained due to serious descent criterion (13). Since $\mathbf{x}_{k+1} = \mathbf{y}_{k+1}$ for serious steps and $\|\mathbf{y}_{k+1} - \mathbf{x}_{k+1}\| \le t_{max} C$ for null steps (by (2) and due to fact that we use the scaled direction vector $\theta_k \mathbf{d}_k$, where $\theta_k = \min\{\, 1, C/\|\mathbf{d}_k\| \,\}$ in line search) the sequence $(\mathbf{y}_k)$ is also bounded. By the local boundedness and the upper semicontinuity of ∂f (see, e.g., [12]), we obtain the boundedness of the sequence $(\boldsymbol{\xi}_k)$.

Let

$$\mathcal{I} = \{1, \ldots, n+2\}.$$

Using the fact that $\boldsymbol{\xi}_k \in \partial f(\mathbf{y}_k)$ for all $k \geq 1$, Lemma 4, and Carathéodory's theorem (see, e.g., [5]), we deduce that there exist vectors $\mathbf{y}^{k,i}$, $\boldsymbol{\xi}^{k,i}$, and numbers $\lambda^{k,i} \geq 0$ and $\tilde{\sigma}_k$ for $i \in \mathcal{I}$ and $k \geq 1$, such that

$$\begin{aligned} &(\tilde{\boldsymbol{\xi}}_k, \tilde{\sigma}_k) = \sum_{i \in \mathcal{I}} \lambda^{k,i} (\boldsymbol{\xi}^{k,i}, \|\mathbf{y}^{k,i} - \mathbf{x}_k\|), \\ &\boldsymbol{\xi}^{k,i} \in \partial f(\mathbf{y}^{k,i}), \qquad \text{and} \\ &\sum_{i \in \mathcal{I}} \lambda^{k,i} = 1, \end{aligned} \tag{24}$$

with

$$(\mathbf{y}^{k,i}, \boldsymbol{\xi}^{k,i}) \in \{(\mathbf{y}_j, \boldsymbol{\xi}_j) \mid j = 1, \ldots, k\}.$$

From the boundedness of $(\mathbf{y}_k)$, we obtain the existence of points $\mathbf{y}_i^*$ $(i \in \mathcal{I})$, and an infinite set $\mathcal{K}_0 \subset \mathcal{K}$ satisfying $(\mathbf{y}^{k,i})_{k \in \mathcal{K}_0} \to \mathbf{y}_i^*$ for $i \in \mathcal{I}$. The boundedness of $(\boldsymbol{\xi}_k)$ and $(\lambda^{k,i})$ gives us the existence of vectors $\boldsymbol{\xi}_i^* \in \partial f(\mathbf{y}_i^*)$, numbers λ_i^* for $i \in \mathcal{I}$, and an infinite set $\mathcal{K}_1 \subset \mathcal{K}_0$ satisfying $(\boldsymbol{\xi}^{k,i})_{k \in \mathcal{K}_1} \to \boldsymbol{\xi}_i^*$ and $(\lambda^{k,i})_{k \in \mathcal{K}_1} \to \lambda_i^*$ for $i \in \mathcal{I}$.

It can be seen from (24) that

$$\lambda_i^* \geq 0 \quad \text{for } i \in \mathcal{I}, \qquad \text{and} \qquad \sum_{i \in \mathcal{I}} \lambda_i^* = 1.$$

From $(w_k)_{k \in \mathcal{K}} \to 0$, Lemma 3, and Lemma 4, we obtain

$$(\tilde{\boldsymbol{\xi}}_k)_{k \in \mathcal{K}} \to \mathbf{0}, \qquad (\tilde{\beta}_k)_{k \in \mathcal{K}} \to 0, \qquad \text{and} \qquad (\tilde{\sigma}_k)_{k \in \mathcal{K}} \to 0.$$

By letting $k \in \mathcal{K}_1$ approach infinity in (24), and by using Lemma 5 with

$$\begin{aligned} &\bar{\mathbf{x}} = \mathbf{x}_k, \qquad l = n+2, \qquad \bar{\mathbf{g}} = \mathbf{0}, \\ &\bar{\boldsymbol{\xi}}_i = \boldsymbol{\xi}_i^*, \qquad \bar{\mathbf{y}}_i = \mathbf{y}_i^*, \qquad \bar{\lambda}_i = \lambda_i^* \end{aligned}$$

we obtain $\mathbf{0} \in \partial f(\bar{\mathbf{x}})$. □

LEMMA 9. *Suppose that there exist vectors* $\mathbf{p}$ *and* $\mathbf{g}$ *together with numbers* $w \geq 0$, $\alpha \geq 0$, $\beta \geq 0$, $M \geq 0$, *and* $c \in (0, 1/2)$ *such that*

$$w = \|\mathbf{p}\|^2 + 2\alpha, \quad \beta + \mathbf{p}^T\mathbf{g} \leq cw, \quad \textit{and} \quad \max\{\|\mathbf{p}\|, \|\mathbf{g}\|, \sqrt{\alpha}\} \leq M.$$

Let $Q : [0,1] \to \mathbb{R}$ *be such that*

$$Q(\lambda) = \|\lambda \mathbf{g} + (1-\lambda)\mathbf{p}\|^2 + 2(\lambda\beta + (1-\lambda)\alpha) \qquad \textit{and}$$
$$b = (1-2c)/4M.$$

Then

$$\min\{Q(\lambda) \mid \lambda \in [0,1]\} \leq w - w^2 b^2.$$

PROOF. See the proof of Lemma 3.5 in [10]. □

LEMMA 10. *Suppose that the level set* $\{\, \mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \leq f(\mathbf{x}_1) \,\}$ *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*

$$\tilde{\boldsymbol{\xi}}_{k+1}^T D_{k+1} \tilde{\boldsymbol{\xi}}_{k+1} \leq \tilde{\boldsymbol{\xi}}_{k+1}^T D_k \tilde{\boldsymbol{\xi}}_{k+1} \qquad \textit{and} \tag{25}$$
$$\mathrm{tr}(D_k) < \frac{3}{2} n \tag{26}$$

for all $k \geq k^*$, *where* $\mathrm{tr}(D_k)$ *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 SR1 update is not used, we have $D_{k+1} = D_k$, and condition (25) is valid with equalities. Otherwise, if $m_k < m_c$, we have

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

where the denominator $\mathbf{u}_k^T (D_k \mathbf{u}_k - \mathbf{s}_k) > 0$ by Lemma 3. Thus, condition (25) is valid in the case $m_k < m_c$. Finally, if $m_k = m_c$, 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 4, Steps 3 and 10). Since $i_{CN} = 0$ for all $k \geq m$, the correction ϱI is not added to new matrix D_{k+1} and, thus, condition (25) is valid also in this case. Furthermore, in each case we have

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

for $k \geq m$, since the matrix $D_k - I$ is negative (semi)definite. Therefore, if $i_{CN} = 0$ for all $k \geq m$, conditions (25) and (26) 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 (25) 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 (25) and (26) are valid for all $k \geq k^*$ with

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

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

LEMMA 11. *Suppose that the level set* $\{ \mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \leq f(\mathbf{x}_1) \}$ *is bounded, the number of serious steps is finite, and the last serious step occurred at the iteration* $m-1$. *Then, the point* $\mathbf{x}_m$ *is stationary for* f.

PROOF. From (10), (11), (12), Lemma 3, and Lemma 10 we obtain

$$\begin{aligned} w_{k+1} &= \tilde{\boldsymbol{\xi}}_{k+1}^T D_{k+1} \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &\leq \tilde{\boldsymbol{\xi}}_{k+1}^T D_k \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &= \varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) \\ &\leq \varphi(0, 0, 1) \\ &= \tilde{\boldsymbol{\xi}}_k^T D_k \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k \\ &= w_k \end{aligned} \tag{27}$$

for $k \geq k^*$ with k^* defined in Lemma 10.

Let us, for a while, denote $D_k = W_k^T W_k$. Thus, 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 (27) we obtain the boundedness of the sequences (w_k), $(W_k\tilde{\boldsymbol{\xi}}_k)$, and $(\tilde{\beta}_k)$. Furthermore, Lemma 10 assures the boundedness of (D_k) and (W_k). (We say that matrix is bounded if its eigenvalues lie in the compact interval that does not contain zero). By Lemma 8, we obtain the boundedness of $(\mathbf{y}_k)$, $(\boldsymbol{\xi}_k)$, and $(W_k\boldsymbol{\xi}_k)$. Let us denote

$$\begin{aligned} &M = \sup\{\|W_k\boldsymbol{\xi}_{k+1}\|, \|W_k\tilde{\boldsymbol{\xi}}_k\|, \sqrt{\tilde{\beta}_k} \mid k \geq k^*\}, \quad \text{and} \\ &b = (1 - 2\varepsilon_R^I)/4M, \end{aligned}$$

and let us first assume that $w_k > \delta > 0$ for all $k \geq k^*$. From the fact that

$$\begin{aligned} &\min\{\varphi(\lambda_1, \lambda_2, \lambda_3) \mid \lambda_i \geq 0,\ i = 1, 2, 3, \sum_{i=1}^{3} \lambda_i = 1\} \\ &\leq \min\{\varphi(0, \lambda, (1 - \lambda)) \mid \lambda \in [0, 1]\} \end{aligned}$$

and (27) we obtain

$$\begin{aligned} w_{k+1} \leq \min\{&\|\lambda W_k\boldsymbol{\xi}_{k+1} + (1 - \lambda)W_k\tilde{\boldsymbol{\xi}}_k)\|^2 + \\ &2(\lambda\beta_{k+1} + (1 - \lambda)\tilde{\beta}_k) \mid \lambda \in [0, 1]\}. \end{aligned}$$

Since $w_k = \tilde{\boldsymbol{\xi}}_k^T D_k\tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k$ by Lemma 3, $\mathbf{d}_k = -D_k\tilde{\boldsymbol{\xi}}_k$ (see Algorithm 1, Step 2), and

$$-\beta_{k+1} + \theta_k\mathbf{d}_k^T\boldsymbol{\xi}_{k+1} \geq -\epsilon_R^k w_k$$

by (4) and due to fact that we use scaled direction vector $\theta_k\mathbf{d}_k$ in line search (see Algorithm 1, Step 5), we have

$$\theta_k\beta_{k+1} - \theta_k\mathbf{d}_k^T\boldsymbol{\xi}_{k+1} \leq \beta_{k+1} - \theta_k\mathbf{d}_k^T\boldsymbol{\xi}_{k+1} \leq \epsilon_R^k w_k = \theta_k\varepsilon_R^I w_k.$$

Now, we can use Lemma 9 with

$$\begin{aligned} &\mathbf{p} = W_k\tilde{\boldsymbol{\xi}}_k, \qquad &&\mathbf{g} = W_k\boldsymbol{\xi}_{k+1}, \qquad &&w = w_k, \\ &\alpha = \tilde{\beta}_k, \qquad &&\beta = \beta_{k+1}, \qquad &&c = \varepsilon_R^I \end{aligned}$$

to obtain

$$w_{k+1} \leq w_k - (w_k b)^2 < w_k - (\delta b)^2$$

for $k \geq k^*$ and, thus, for sufficiently large k, we have a contradiction with the assumption $w_k > \delta$. Therefore, due to monotonicity of w_k for $k \geq k^*$, we obtain $w_k \to 0$, $\mathbf{x}_k \to \mathbf{x}_m$, and by Lemma 8 we have $\mathbf{0} \in f(\mathbf{x}_m)$. □

THEOREM 12. *Suppose that the level set* $\{ \mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \leq f(\mathbf{x}_1) \}$ *is bounded, then every cluster point of sequence* $(\mathbf{x}_k)$ *is stationary for* f.

PROOF. Let $\bar{\mathbf{x}}$ be a cluster point of $(\mathbf{x}_k)$, and let $\mathcal{K} \subset \{1, 2, \ldots\}$ be an infinite set such that $(\mathbf{x}_k)_{k \in \mathcal{K}} \to \bar{\mathbf{x}}$. In view of Lemma 11, we can restrict the consideration to the case where the number of serious steps (with $t_L^k > 0$) is infinite. We denote

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

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

$$0 \leq \varepsilon_L^k t_L^k w_k \leq f_k - f_{k+1} \to 0 \qquad \text{for } k \geq 1. \tag{28}$$

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 $(\mathbf{x}_k)_{k \in \mathcal{K}_1} \to \bar{\mathbf{x}}$ by (28) and, thus, by Lemma 8 we have $\mathbf{0} \in f(\bar{\mathbf{x}})$.

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

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

for all $k \geq 1$. Thus, $(\|\mathbf{x}_{k+1} - \mathbf{x}_k\|)_{k \in \mathcal{K}_2} \to 0$. By (9), (28), and the boundedness of $(\boldsymbol{\xi}_k)$ by Lemma 8, and since $\mathbf{y}_{k+1} = \mathbf{x}_{k+1}$ 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 $(w_k)_{k \in \mathcal{K}_3} \to 0$, $(\mathbf{x}_k)_{k \in \mathcal{K}_3} \to \bar{\mathbf{x}}$, and $\mathbf{0} \in f(\bar{\mathbf{x}})$ by Lemma 8. □

Note that, if we choose $\varepsilon > 0$, Algorithm 1 terminates in a finite number of steps, since $(w_k) \to 0$ and $(q_k) \to 0$ in case the number of serious steps is finite (see the proof of Lemma 11), and 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 $(q_k)_{k \in \mathcal{K}_3} \to 0$ in case the number of serious steps is infinite (see the proof of Theorem 12).

4 Conclusion

In this paper, we have introduced a limited memory bundle method for nonsmooth large-scale optimization. We have proved the global convergence of the method for locally Lipschitz continuous objective functions, which are not necessarily differentiable nor convex.

Although the new method is quite useful already, there is a lot of 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, an alternative ways of scaling and skipping the updates (especially, the SR1 update), constraint handling (simple bounds, linear constraints, nonlinear constraints), and parallelized version of the program.

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 COMAS Graduate School of the University of Jyväskylä, TEKES and Academy of Finland grant #65760.

References

[1] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.

[2] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley-Interscience, New York, 1983.

[3] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, Chichester, second edition, 1987.

[4] M. Haarala, K. Miettinen, and M. M. Mäkelä. Large-scale nonsmooth optimization: New variable metric bundle algorithm with limited memory.

Reports of the Department of Mathematical Information Technology, Series B. Scientific computing, B 6/2003 University of Jyväskylä, Jyväskylä, 2003.

[5] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I.* Springer-Verlag, Berlin, 1993.

[6] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II.* Springer-Verlag, Berlin, 1993.

[7] K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization.* Lecture Notes in Mathematics 1133. Springer-Verlag, Berlin, 1985.

[8] T. Kärkkäinen, K. Majava, and M. M. Mäkelä. Comparison of formulations and solution methods for image restoration problems. *Inverse Problems*, 17(6):1977–1995, 2001.

[9] C. Lemaréchal. Nondifferentiable optimization. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, pages 529–572. North-Holland, Amsterdam, 1989.

[10] L. Lukšan and J. Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102:593–613, 1999.

[11] M. M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17(1):1–29, 2002.

[12] M. M. Mäkelä and P. Neittaanmäki. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control.* World Scientific Publishing Co., Singapore, 1992.

[13] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[14] J. Vlček and L. Lukšan. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications*, 111(2):407–430, 2001.