LIMITED MEMORY BUNDLE METHOD FOR LARGE BOUND CONSTRAINED NONSMOOTH OPTIMIZATION: CONVERGENCE ANALYSIS

Napsu Karmitsa[1] Marko M. Mäkelä[2]

Department of Mathematics, University of Turku,
FI-20014 Turku, Finland.

Abstract: Practical optimization problems often involve nonsmooth functions of hundreds or thousands of variables. As a rule, the variables in such large problems are restricted to certain meaningful intervals. In the paper [Karmitsa, Mäkelä, 2009] we described an efficient limited memory bundle method for large-scale nonsmooth, possibly nonconvex, bound constrained optimization. Although this method works very well in numerical experiments, it suffers from one theoretical drawback, namely that it is not necessarily globally convergent. In this paper, a new variant of the method is proposed and its global convergence for locally Lipschitz continuous functions is proved.

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

1 Introduction

In this paper, a global convergence theory is provided for a new version of limited memory bundle algorithm LMBM-B [19] for solving large nonsmooth (nondifferentiable) bound constrained optimization problems. These kinds of problems frequently appear, for instance, in many areas of industrial design, process control, and economic planning (see e.g. [6, 28, 29, 31]), and, due to nonsmoothness, they are difficult or even impossible to solve using classical gradient-based optimization methods. On the other hand, none of the current general nonsmooth optimization solvers (especially, those capable of constraint handling) can be used efficiently when the number of variables is large, say 1000 or more. This means that there is an evident need for methods able to solve large, possibly nonconvex, nonsmooth, (bound) constrained optimization problems.

We consider the problem

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x}^l \le \boldsymbol{x} \le \boldsymbol{x}^u, \end{cases} \tag{1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous and the number of variables n is large. Moreover, the vectors $\boldsymbol{x}^l$ and $\boldsymbol{x}^u$ representing the lower and the upper bounds on the variables are fixed and the inequalities in (1) are taken component-wise. A point $\boldsymbol{x} \in \mathbb{R}^n$ satisfying the bounds is called *feasible* and

[1] Corresponding author, E-mail: napsu@karmitsa.fi

[2] E-mail: makela@utu.fi

a set $\mathcal{F}$ of all feasible points is called the *feasible region* for problem (1). That is, $\mathcal{F} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^l \leq \boldsymbol{x} \leq \boldsymbol{x}^u\}$.

Nowadays different variants of bundle methods (see e.g. [16, 20, 24, 26, 33]) are recognized as the most effective and reliable methods for solving nonsmooth optimization problems. The basic assumption for these methods is that at every point $\boldsymbol{x} \in \mathbb{R}^n$, we can evaluate the objective function $f(\boldsymbol{x})$ and an arbitrary subgradient $\boldsymbol{\xi} \in \mathbb{R}^n$ from the subdifferential [5]

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

where “conv” denotes the convex hull of a set. The idea of bundle methods is to approximate the subdifferential or, to be exact, the Goldstein ε-subdifferential (see e.g. [26])

$$\partial_\varepsilon^G f(\boldsymbol{x}) = \operatorname{conv}\{ \partial f(\boldsymbol{y}) \mid \boldsymbol{y} \in \bar{B}(\boldsymbol{x};\varepsilon) \}$$

by gathering subgradients from previous iterations into a bundle. Here, we have $\boldsymbol{y} \in \mathbb{R}^n$ and $\varepsilon \geq 0$, and $\bar{B}(\boldsymbol{x};\varepsilon)$ denotes a closed ball with center $\boldsymbol{x}$ and radius ε. Note that $\partial f(\boldsymbol{x}) \subseteq \partial_\varepsilon^G f(\boldsymbol{x})$ for all $\varepsilon \geq 0$.

While standard bundle methods are very efficient for small- and medium-scale problems, they are not, in general, competent in large-scale settings (see e.g. [1, 13, 17]). In [12, 13, 14] we have proposed a limited memory bundle method (LMBM) for general, possibly nonconvex, nonsmooth large-scale unconstrained optimization. The idea of LMBM is to combine the variable metric bundle methods [23, 34] for small- and medium-scale nonsmooth optimization with the limited memory variable metric methods (see e.g. [4, 11, 22, 30]) for large-scale smooth optimization. LMBM exploits the ideas of the variable metric bundle methods, namely the utilization of null steps and simple aggregation of subgradients, but the search direction is calculated using a limited memory approach. Therefore, the time-consuming quadratic direction finding problem which appears in standard bundle methods (see e.g. [20, 26, 33]) need not be solved and the number of stored subgradients (i.e. the size of the bundle) is independent of the dimension of the problem. Furthermore, LMBM uses only a 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 [23, 34].

In [19], a new variant of the method, LMBM-B, suitable for solving bound constrained problems, was introduced. In LMBM-B the constraint handling is based on subgradient projection and dual subspace minimization and it is adapted from the smooth limited memory BFGS method for bound constrained optimization [3]. Although numerically very efficient, the method described in [19] suffers from the theoretical drawback that it is not necessarily globally convergent in the nonsmooth case.

In order to prove the global convergence of the new variant of LMBM-B , some modifications had to be made to the algorithm introduced in [19]. We included so-called *stark projections of subgradients* in the aggregation procedure to guarantee the convergence of aggregate subgradients to zero. Moreover, we added some *corrections to the limited memory matrices* to preserve sufficient positive definiteness and boundedness of these matrices whenever necessary, and finally we adopted a slightly *modified line search procedure*. Although this may sound like an easy task, several open questions had to be answered and many implementational challenges had to be overcome before obtaining a new version which could boast even a hint of the efficiency of the previous version and which preserves the necessary convergence properties.

The rest of this paper is organized as follows. In Section 2, we describe the new version of the algorithm LMBM-B for bound constrained optimization. We start by giving a brief introduction to the method. After that, we describe in detail the direction finding procedure, usage of serious and null steps, as well as the aggregation procedure. We combine these details in the model algorithm. We also give a special line search procedure, which is modified from that used in [14, 19]. In Section 3, we prove the global convergence of the method for locally Lipschitz continuous objective functions that are not necessarily differentiable or convex. Some preliminary results of numerical experiments are presented in Section 4 and, finally, in Section 5, we conclude the paper. A detailed description of limited memory matrix updating, conditions for limited memory matrices to preserve positive definiteness, and some details of the computation of corrected matrices are given in the Appendix.

2 Method

The globally convergent version of LMBM-B (see Figure 1) is characterized by the use of null steps together with the aggregation and projection of subgradients. Moreover, the limited memory approach is utilized in the calculation of the search direction and the aggregate values. The use of null steps gives further information about the nonsmooth objective function in the case when the search direction is not "good enough". On the other hand, simple aggregation and stark projection of subgradients guarantee the convergence of projected aggregate subgradients to zero and make it possible to evaluate a termination criterion.

Direction finding. The search direction is calculated using a two-stage approach. First, we define the quadratic model function q_k that approximates the objective function at the iteration point $\boldsymbol{x}_k$ by

$$q_k(\boldsymbol{x}) = f(\boldsymbol{x}_k) + \tilde{\boldsymbol{\xi}}_k^T(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T B_k(\boldsymbol{x} - \boldsymbol{x}_k). \tag{2}$$

Here $\tilde{\boldsymbol{\xi}}_k$ is the aggregate subgradient of the objective function from the previous iteration and B_k is a positive definite limited memory variable metric update that, in the smooth case, represents the approximation of the Hessian matrix. Now, the generalized gradient projection method is used to find the generalized Cauchy point $\boldsymbol{x}_k^c$ [7] and, at the same time, to identify the active set $\mathcal{I}_A^k$ of the problem. The procedure used in LMBM-B is based on that in [3]. However, in LMBM-B we use the aggregate subgradient of the objective function instead of the gradient and, in addition to the limited memory BFGS update formula, we utilize the limited memory SR1 update whenever necessary, that is, after a null step (see Figure 1).

The generalized Cauchy point at iteration k is defined as the first local minimizer (starting from $\boldsymbol{x}_k$) of the univariate piecewise quadratic function

$$\hat{q}_k(t) = q_k(\mathcal{P}_c[\boldsymbol{x}_k - t\tilde{\boldsymbol{\xi}}_k, \boldsymbol{x}^l, \boldsymbol{x}^u])$$

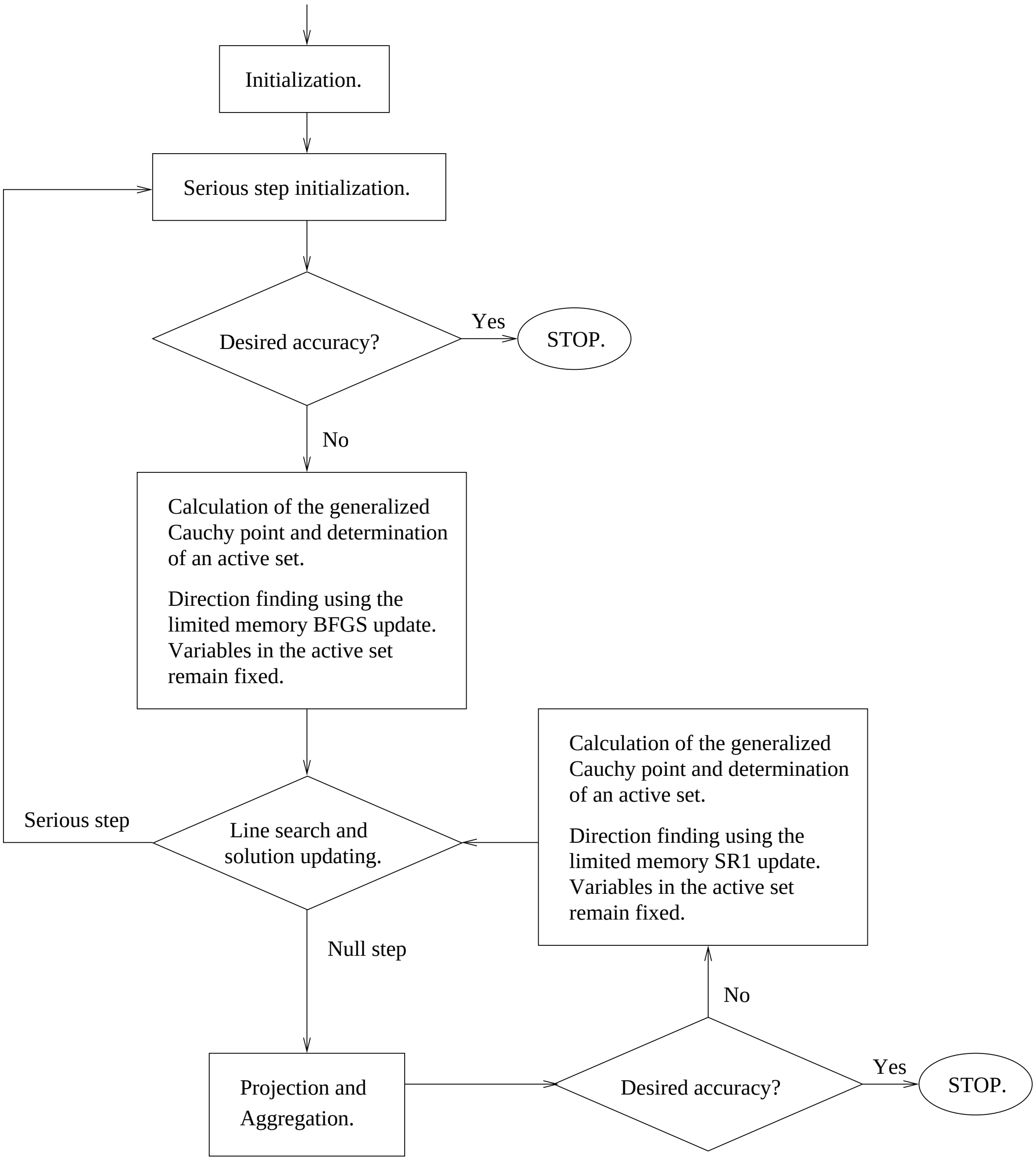

Initialization.
Serious step initialization.
Desired accuracy?
Yes
STOP.
No
Calculation of the generalized Cauchy point and determination of an active set.
Direction finding using the limited memory BFGS update. Variables in the active set remain fixed.
Serious step
Line search and solution updating.
Calculation of the generalized Cauchy point and determination of an active set.
Direction finding using the limited memory SR1 update. Variables in the active set remain fixed.
Null step
No
Projection and Aggregation.
Desired accuracy?
Yes
STOP.

Figure 1: Globally convergent version of LMBM-B.

along the projected gradient direction $\mathcal{P}_c[\boldsymbol{x}_k - t\tilde{\boldsymbol{\xi}}_k, \boldsymbol{x}^l, \boldsymbol{x}^u] - \boldsymbol{x}_k$ (see e.g. [7]). Here we have defined the projection operator $\mathcal{P}_c[\cdot]$ (component-wise) by

$$\mathcal{P}_c[\boldsymbol{x}, \boldsymbol{x}^l, \boldsymbol{x}^u]_i = \begin{cases} x_i^l, & \text{if } x_i < x_i^l \\ x_i, & \text{if } x_i \in [x_i^l, x_i^u] \\ x_i^u, & \text{if } x_i > x_i^u. \end{cases}$$

This operator projects the point $\boldsymbol{x}$ into the feasible region $\mathcal{F}$ defined by the bounds $\boldsymbol{x}^l$ and $\boldsymbol{x}^u$. Now, if we denote by t_k^c the value of t corresponding to the first local minimum of $\hat{q}_k(t)$, then the generalized Cauchy point is given by

$$\boldsymbol{x}_k^c = \mathcal{P}_c[\boldsymbol{x}_k - t_k^c\tilde{\boldsymbol{\xi}}_k, \boldsymbol{x}^l, \boldsymbol{x}^u]. \tag{3}$$

The variables whose values at $\boldsymbol{x}_k^c$ are at the lower or upper bound comprise the active set $\mathcal{I}_A^k = \{i \mid x_{k,i}^c = x_i^l \text{ or } x_{k,i}^c = x_i^u\}$, where we have denoted by $x_{k,i}^c$ the ith component of the vector $\boldsymbol{x}_k^c$. The calculation of this generalized Cauchy point makes it possible to add and delete several bounds from the active set during a single iteration, which may be an important feature for both nonsmooth [32] and large-scale [8] problems. For details of the practical computation of the generalized Cauchy point, see [3, 18, 19].

When the generalized Cauchy point has been found, we approximately minimize the quadratic model function (2) over the space of free variables, in other words, the variables in the active set are treated as equality constraints. The subspace minimization procedure used is, in principal, the same as the dual space method in [3]. But, as before, we use the aggregate subgradient of the objective function and employ the limited memory SR1 update if the previous step taken was a null step.

We obtain the search direction $\boldsymbol{d}$ by solving the smooth quadratic problem

$$\begin{cases} \text{minimize} & \tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T B_k \boldsymbol{d} \\ \text{such that} & A_k^T \boldsymbol{d} = \boldsymbol{b}_k, \\ & \boldsymbol{x}^l \leq \boldsymbol{x}_k + \boldsymbol{d} \leq \boldsymbol{x}^u, \end{cases} \tag{4}$$

where A_k is the matrix of active constraints gradients at $\boldsymbol{x}_k^c$ and $\boldsymbol{b}_k = A_k^T(\boldsymbol{x}_k^c - \boldsymbol{x}_k)$. Note that A_k consists of n_A unit vectors (where n_A is the number of elements in the active set $\mathcal{I}_A^k$) and $A_k^T A_k$ is equal to the identity.

We first ignore the bound constraints. The first order sufficient optimality conditions for problem (4) without bounds are

$$\tilde{\boldsymbol{\xi}}_k + B_k \boldsymbol{d}_k^* + A_k \boldsymbol{\mu}_k^* = \boldsymbol{0} \tag{5}$$

$$A_k^T \boldsymbol{d}_k^* = \boldsymbol{b}_k, \tag{6}$$

where $\boldsymbol{\mu}_k^* \in \mathbb{R}^{n_A}$ are the Lagrange multipliers of problem (4).

In what follows, we denote by D_k the update formula which is the inverse of B_k. Now, by multiplying (5) by $A_k^T D_k$ and by using (6), we obtain

$$(A_k^T D_k A_k)\boldsymbol{\mu}_k^* = -A_k^T D_k \tilde{\boldsymbol{\xi}}_k - \boldsymbol{b}_k. \tag{7}$$

The linear system (7) can be solved by using the Sherman-Morrison-Woodbury formula and the compact representation of limited memory matrices (see [3]). Thus, $\boldsymbol{d}_k^*$ is given by

$$\boldsymbol{d}_k^* = -D_k(A_k \boldsymbol{\mu}_k^* + \tilde{\boldsymbol{\xi}}_k). \tag{8}$$

If there are no active variables, we simply obtain $\boldsymbol{d}_k^* = -D_k \tilde{\boldsymbol{\xi}}_k$, which is the formula used in the original unconstrained version of the limited memory bundle method [12, 13, 14]. In the case where the vector $\boldsymbol{x}_k + \boldsymbol{d}_k^*$ violates the bounds in (4), we backtrack, as in [3], along the line joining the infeasible point $\boldsymbol{x}_k + \boldsymbol{d}_k^*$ and the generalized Cauchy point $\boldsymbol{x}_k^c$ to regain the feasible region. That is, we first compute

$$\alpha_k^* = \min\left\{1, \max\{\alpha \mid x_i^l \leq x_{k,i}^c + \alpha(x_{k,i} + d_i^* - x_{k,i}^c) \leq x_i^u,\ i \in \mathcal{I}_F^k\}\right\}, \tag{9}$$

where $\mathcal{I}_F^k = \{j \mid j = \{1, \ldots, n\} \setminus \mathcal{I}_A^k\}$ is the set of free variables, and then, we set $\bar{\boldsymbol{x}} = \boldsymbol{x}_k^c + \alpha_k^*(\boldsymbol{x}_k + \boldsymbol{d}_k^* - \boldsymbol{x}_k^c)$ and $\boldsymbol{d}_k = \bar{\boldsymbol{x}} - \boldsymbol{x}_k$. Again, see [3, 18, 19] for details of the practical computations.

Solution updating. LMBM-B generates a sequence of basic points $\{\boldsymbol{x}_k\} \subset \mathcal{F}$ together with a sequence of auxiliary points $\{\boldsymbol{y}_k\} \subset \mathcal{F}$. A new auxiliary point $\boldsymbol{y}_{k+1}$ is produced using a special line search procedure such that

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t^k \boldsymbol{d}_k, \qquad \text{for } k \geq 1 \tag{10}$$

with $\boldsymbol{y}_1 = \boldsymbol{x}_1$, where $t^k \in (0, t^k_{max}]$ is a step size and $t^k_{max} \geq 1$ is the upper bound for the step size selected such that it assures the feasibility of the produced points.

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

$$f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L t^k w_k, \tag{11}$$

where $\varepsilon_L \in (0, 1/2)$ is a line search parameter and $w_k > 0$ represents the desirable amount of descent of f at $\boldsymbol{x}_k$. If condition (11) is satisfied, we set

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$$

and a serious step is taken.

Otherwise, we take *a null step*. In this case, the use of the special line search procedure guarantees that we have

$$-\beta_{k+1} - \mathcal{P}_{\boldsymbol{x}_k}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_k}[\boldsymbol{\xi}_{k+1}] \geq -\varepsilon_R w_k, \tag{12}$$

where $\varepsilon_R \in (\varepsilon_L, 1/2)$ is a line search parameter, $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, $\mathcal{P}_{\boldsymbol{x}}[\boldsymbol{\xi}]$ denotes a stark projection of $\boldsymbol{\xi}$ at $\boldsymbol{x}$ (to be defined shortly), and β_{k+1} is the subgradient locality measure [21, 27] similar to that used in 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 enriched because we store the auxiliary point $\boldsymbol{y}_{k+1}$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1}$.

Aggregation. LMBM-B uses the original subgradient $\boldsymbol{\xi}_k \in \partial f(\boldsymbol{x}_k)$ after the serious step and the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$ after the null step for direction finding (i.e. we set $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k$ if the previous step was a serious step). The aggregation procedure used in the previous versions of LMBM [12, 13, 14, 19] is similar to that of the original variable metric bundle methods [23, 34], except that the variable metric updates are calculated using the limited memory approach. However, in order to guarantee the global convergence of the bound constrained version, we need to consider projections of subgradients instead of original subgradients in the aggregation procedure. It may seem that the use of the active set $\mathcal{I}^k_A$ in the projection procedure would be a natural choice. However, the active set $\mathcal{I}^k_A$ is calculated at the generalized Cauchy point $\boldsymbol{x}^c_k$ and it may change over consecutive null steps, which is highly undesirable from the viewpoint of global convergence. Therefore, we instead calculate a very simple projection at point $\boldsymbol{x}_k$ that does not change ($\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ in null steps). We define this stark projection operator $\mathcal{P}_{\boldsymbol{x}}[\cdot]$ at point $\boldsymbol{x}$ (component-wise) by

$$\mathcal{P}_{\boldsymbol{x}}[\boldsymbol{\xi}]_i = \begin{cases} 0, & \text{if } x^l_i - x_i \geq 0 \\ \xi_i, & \text{if } x_i \in (x^l_i, x^u_i) \\ 0, & \text{if } x^u_i - x_i \leq 0. \end{cases} \tag{13}$$

In what follows, we call this projection the $\mathcal{P}_{\boldsymbol{x}}$-projection (at point $\boldsymbol{x}$).

Now, the aggregation procedure is carried out by determining multipliers λ_i^k satisfying $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^{3} \lambda_i^k = 1$ that minimize the function

$$\begin{aligned}\varphi(\lambda_1, \lambda_2, \lambda_3) = \mathcal{P}_{\boldsymbol{x}_k}[\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k\,]^T D_k \mathcal{P}_{\boldsymbol{x}_k}[\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k\,] \\ + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k). \qquad (14)\end{aligned}$$

Here $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$ is the current subgradient, $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ is the auxiliary subgradient, and $\tilde{\boldsymbol{\xi}}_k$ is the current aggregate subgradient from the previous iteration ($\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$). In addition, β_{k+1} is the current locality measure and $\tilde{\beta}_k$ is the current aggregate locality measure ($\tilde{\beta}_1 = 0$). In the above, we denote by m the index of the iteration after the latest serious step, i.e. $\boldsymbol{x}_k = \boldsymbol{x}_m$. Note that we do not use index k (or subgradient $\boldsymbol{\xi}_k$) instead of m (or $\boldsymbol{\xi}_m$), since $\boldsymbol{\xi}_k \in \partial f(\boldsymbol{y}_k)$ and, in the case of consecutive null steps, $\boldsymbol{y}_k$ is not equal to $\boldsymbol{x}_k$. In the algorithm, index m is set equal to k after each serious step but it is not updated after a null step (see Step 1 in Algorithm 2.1).

The next $\tilde{\boldsymbol{\xi}}_{k+1}$ is defined as a convex combination of the subgradients mentioned above:

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad (15)$$

and the next $\tilde{\beta}_{k+1}$ as a convex combination of the locality measures:

$$\tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \qquad (16)$$

The $\mathcal{P}_{\boldsymbol{x}}$-projection depends only on point $\boldsymbol{x}$ and not on the subgradient $\boldsymbol{\xi}$ in question. Thus, $\mathcal{P}_{\boldsymbol{x}_k}[\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k] = \lambda_1 \mathcal{P}_{\boldsymbol{x}_k}[\boldsymbol{\xi}_m\,] + \lambda_2 \mathcal{P}_{\boldsymbol{x}_k}[\boldsymbol{\xi}_{k+1}\,] + \lambda_3 \mathcal{P}_{\boldsymbol{x}_k}[\tilde{\boldsymbol{\xi}}_k\,]$, making the minimization of function (14) rather an easy task.

Matrix updating. We use the limited memory approach (see e.g. [4, 30] and the Appendix) in the calculation of the generalized Cauchy point, search direction, and aggregate values. The idea of limited memory matrix updating is that, instead of storing the large $n \times n$ -matrices B_k and D_k, one stores a certain (usually small constant) number $\hat{m}_c$ of vectors, so-called correction pairs obtained at the previous iterations of the algorithm, and uses these correction pairs to implicitly define the variable metric matrices. When the storage space available is used up, the oldest correction pair is deleted to make room for a new one; thus, except for the first few iterations, we always have the $\hat{m}_c$ most recent correction pairs available.

The utilization of the limited memory approach means, of course, that the variable metric updates are not as accurate as if we used standard variable metric updates (see e.g. [10]). However, both the storage space required and the number of operations needed in the calculations are significantly smaller. Indeed, the number of operations needed is $O(n)$ while with standard variable metric updates used in original variable metric bundle methods [23, 34], $O(n^2)$ operations are needed.

The limited memory variable metric matrices used in our algorithm are represented in the compact matrix form originally described in [4]. We use both the limited memory BFGS and the limited memory SR1 update formulae in the calculations of the search direction and the aggregate values. If the previous step was a null step, the matrices D_k and B_k are formed using the limited memory SR1 updates (see equations (27) and (28) in the Appendix). The SR1 update formulae allows us to preserve the boundedness and some other good properties of the generated matrices which guarantee the global convergence of the method. Otherwise, since these properties are not required after a serious step, the more efficient limited memory BFGS updates (see equations (25) and (26) in the Appendix) are employed.

The SR1-update formulae do not in general preserve positive definiteness. Moreover, both the nonconvexity and bounds may violate the condition $\boldsymbol{d}_i^T(\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_m) > 0$ for all $i = 1, ..., k-1$, which is the classical condition for the positive definiteness of the BFGS update. Thus, to maintain the positive definiteness of the generated matrices, we simply skip the individual updates (discard the newest correction pair rather than the oldest one) if they violate positive definiteness (for more details, see the Appendix).

The basic assumption for bundle methods to converge is that after a null step we have $\boldsymbol{z}^T D_{k+1} \boldsymbol{z} \leq \boldsymbol{z}^T D_k \boldsymbol{z}$ for all $\boldsymbol{z} \in \mathbb{R}^n$. In LMBM-B this is guaranteed by the special limited memory SR1 update [12, 14]. In addition, to ensure the global convergence of the method, the matrices D_k are assumed to be uniformly positive definite and uniformly bounded (we say that a matrix is bounded if its eigenvalues lie in a compact interval that does not contain zero). This is guaranteed by adding a positive definite correction matrix to matrix D_k when necessary (i.e. we set $D_k = D_k + \sigma I$ with $\sigma > 0$).

Algorithm. We now present the algorithm for bound constrained nonsmooth optimization. After that, we show how to select the appropriate step size. In what follows, we assume that at every feasible point $\boldsymbol{x} \in \mathcal{F} \subset \mathbb{R}^n$ we can evaluate the objective function $f(\boldsymbol{x})$ and a subgradient $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$.

Algorithm 2.1. (Globally convergent version of LMBM-B)

Data: Choose the final accuracy tolerance $\varepsilon > 0$, the positive line search parameters $\varepsilon_L \in (0, 1/2)$ and $\varepsilon_R \in (\varepsilon_L, 1/2)$, and the distance measure parameter $\gamma \geq 0$ (with $\gamma = 0$ if f is convex). Select the lower and upper bounds $t_{min} \in (0, 1)$ and $t_{max} \geq 1$ for the step size. Select the control parameter $C > 0$ for the length of the direction vector and a correction parameter $\sigma \in (0, 1/2)$. Select an upper limit $\hat{m}_c \geq 3$ for the number of stored correction pairs.

Step 0: (*Initialization.*) Choose a (feasible) starting point $\boldsymbol{x}_1 \in \mathcal{F} \subset \mathbb{R}^n$. Initialize the limited memory matrices $S_1 = U_1 = [\,]$ (empty matrices) and the scaling parameter $\vartheta_1 = 1$. 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 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 update the index for the serious step $m = k$.

Step 2: (*Stopping criterion and correction.*) Calculate the values $\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]$ and $\sigma \|\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]\|^2$ using the stark projection (13). If $m = k$, use the limited memory BFGS update formula (25) for the calculation of D_k. Otherwise, use the limited memory SR1 update formula (27). If $\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k] \leq \sigma \|\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]\|^2$ or $i_{CN} = 1$, set

$$w_k = \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k] + \sigma \|\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]\|^2 + 2\tilde{\beta}_k \tag{17}$$

(i.e. $D_k = D_k + \sigma I$) and $i_{CN} = 1$. Otherwise, set

$$w_k = \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k] + 2\tilde{\beta}_k. \tag{18}$$

If $w_k \leq \varepsilon$, $\xi_{k,i} \leq 0$ for all i such that $x_{k,i} = x_i^u$ and $\xi_{k,i} \geq 0$ for all i such that $x_{k,i} = x_i^l$, then stop with $\boldsymbol{x}_k$ as the final solution.

Step 3: (*Generalized Cauchy point.*) Compute the generalized Cauchy point $\boldsymbol{x}_k^c$ (see (3)) and determine the active set $\mathcal{I}_A^k = \{\, i \mid x_{k,i}^c = x_i^l \text{ or } x_{k,i}^c = x_i^u \}$. Use the same type of update formula for B_k as in Step 2 for D_k. Note that $B_k = D_k^{-1}$ if $i_{CN} = 0$ and $B_k = (D_k + \sigma I)^{-1}$, otherwise.

Step 4: (*Direction finding.*) Compute the search direction $\boldsymbol{d}_k$ (see (8)). Use the same update formula for D_k as in Step 2 and set $D_k = D_k + \sigma I$ if $i_{CN} = 1$. Backtrack if necessary (see (9)). The variables in the active set $\mathcal{I}_A^k$ remain fixed.

Step 5: (*Line search and solution updating.*) Set the scaling parameter for the length of the direction vector and for line search $\theta_k = \min\{\,1, C/\|\boldsymbol{d}_k\|\,\}$. Determine the maximum step size $t_{max}^k \leq t_{max}$ such that $\boldsymbol{x}_k + t_{max}^k \theta_k \boldsymbol{d}_k$ is feasible. Choose the initial step size $t_I^k \in [t_{min}, t_{max}^k]$. Determine the step size $t^k \in (0, t_I^k]$ by Algorithm 2.2. Set the corresponding values

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t^k \theta_k \boldsymbol{d}_k \qquad \text{and} \qquad \boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1}).$$

Set $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ and $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k = t^k \theta_k \boldsymbol{d}_k$ and update the limited memory matrices U_{k+1} and S_{k+1}.

If condition (11) is valid (i.e. we take a serious step), set

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}, \qquad f_{k+1} = f(\boldsymbol{x}_{k+1}), \qquad \text{and} \qquad \beta_{k+1} = 0.$$

Also set $k = k + 1$ and go to Step 1.

Otherwise (i.e. if condition (12) is valid), set

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k \qquad \text{and} \qquad f_{k+1} = f_k,$$

and calculate the locality measure

$$\beta_{k+1} = \max\{|f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + (\boldsymbol{s}_k)^T \boldsymbol{\xi}_{k+1})|, \gamma \|\boldsymbol{s}_k\|^2\,\}. \tag{19}$$

Step 6: (*Aggregation.*) Determine multipliers λ_i^k satisfying $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function (14), where D_k is again calculated using the updating formula as in Step 2 and $D_k = D_k + \sigma I$ if $i_{CN} = 1$. Set

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

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

Note that in Steps 2, 3, 4 and 6, matrices D_k and B_k are not formed explicitly but the limited memory expressions (see the Appendix) are used instead.

Line Search Procedure. Next we consider how to choose the appropriate step size $t^k \in (0, t_I^k]$ in LMBM-B. The initial step size $t_I^k \in [t_{min}, t_{max}^k]$ (see Step 5 in Algorithm 2.1) is selected by using a bundle containing auxiliary points and corresponding function values and subgradients (for the details of this procedure see [34]). Since the aggregation procedure (see Step 6 in Algorithm 2.1) uses only three subgradients, the minimum size of the bundle (m_ξ) is two and a larger bundle (if it is employed) is used only for the selection of the initial step size.

The line search procedure used to determine step size t^k is rather similar to that given in [12, 14] which, in turn, was derived from [34]. However, in order to guarantee global convergence also in the bound constrained case, it was necessary to modify the line search procedure as well as the semi-smoothness assumption used in the previous variants of LMBM.

Algorithm 2.2. *(Modified line search for bound constrained problems).*

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 current vector $\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]^T D_k$ available. 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 $\gamma \geq 0$, the desirable amount of descent w_k, and the positive line search parameters $\varepsilon_L \in (0,1/2)$, $\varepsilon_R \in (\varepsilon_L, 1/2)$, $\varepsilon_A \in (0, \varepsilon_R - \varepsilon_L)$, and $\varepsilon_T \in (\varepsilon_L, \varepsilon_R - \varepsilon_A)$ available. In addition, suppose that we have the number of consecutive null steps $i_{null} \geq 0$ and the maximum number of additional interpolations i_{max}.

Step 0: (*Initialization.*) Set $t_A = 0$, $t = t_U = t_I^k$, and $i_I = 0$, and calculate the interpolation parameter

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

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

$$\beta = \max\{\, |f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) + t\theta_k \boldsymbol{d}_k^T \boldsymbol{\xi}|,\, \gamma\,(t\theta_k \|\boldsymbol{d}_k\|)^2\,\}.$$

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

Step 2: (*Serious step.*) If

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

and either

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

then set $t^k = t$ 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

$$-\beta - \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\boldsymbol{\xi}\,] \geq -\varepsilon_R w_k,$$

then set $t^k = t$ 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.

The line search algorithm 2.2 terminates in a finite number of iterations (the proof is rather similar to that given in [34]) if the problem satisfies the following modified semi-smoothness assumption: For all $\boldsymbol{\xi} \in \partial_\varepsilon^G f(\boldsymbol{x})$ with some small $\varepsilon \geq 0$ and for any $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{d} \in \mathbb{R}^n$, and sequences $\{\hat{\boldsymbol{\xi}}_j\} \subset \mathbb{R}^n$ and $\{t_j\} \subset \mathbb{R}_+$ satisfying $\hat{\boldsymbol{\xi}}_j \in \partial f(\boldsymbol{x}+t_j\boldsymbol{d})$ and $t_j \downarrow 0$, we have

$$-\limsup_{j\to\infty} \mathcal{P}_{\boldsymbol{x}}[\boldsymbol{\xi}\,]^T D \mathcal{P}_{\boldsymbol{x}}[\hat{\boldsymbol{\xi}}_j\,] \geq \liminf_{j\to\infty} \frac{f(\boldsymbol{x}+t_j\boldsymbol{d})-f(\boldsymbol{x})}{t_j}, \tag{20}$$

where $\mathcal{P}_{\boldsymbol{x}}[\cdot]$ denotes the stark projection at point $\boldsymbol{x}$ (see (13)) and D is the inverse variable metric approximation calculated at this very same point. Note that if none of the variables is on the boundary at $\boldsymbol{x}$, this modified semi-smoothness assumption is very similar to the classical semi-smoothness assumption [2].

3 Convergence Analysis

In this section, we prove the global convergence of Algorithm 2.1. That is, we prove that the algorithm converges from any feasible starting point $\boldsymbol{x}_1 \in \mathcal{F}$.

The main differences between the algorithm proposed here and the previous version [19] are in the use of stark projections of subgradients in the aggregation procedure and in the calculation of the stopping criterion (see Steps 2 and 6 of Algorithm 2.1). Moreover, here we use the corrections for the limited memory matrices whenever necessary (see Step 2 of Algorithm 2.1) and keep the length of the direction vector bounded (see Step 5 of Algorithm 2.1). Also the line search procedure (see Algorithm 2.2) has been changed.

We now recall a necessary KKT-type optimality condition for locally Lipschitz continuous objective functions in bound the constrained case (i.e. for problem (1)). Assuming the convexity of the objective function, this condition is also sufficient and the minimum is global (see e.g. [5]).

Theorem 3.1. (KKT-type optimality condition for bound constrained problems.) *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $\boldsymbol{x} \in \mathbb{R}^n$ and let us use the notation $\boldsymbol{g}^l(\boldsymbol{x}) = \boldsymbol{x}^l - \boldsymbol{x}$ and $\boldsymbol{g}^u(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}^u$. If $\boldsymbol{x}$ is a local minimum of problem (1), then there exist Lagrange multipliers $\mu_i^l, \mu_i^u \geq 0$ such that $\mu_i^l g_i^l(\boldsymbol{x}) = 0$ and $\mu_i^u g_i^u(\boldsymbol{x}) = 0$ for all $i \in \{1,\dots,n\}$ and*

$$\mathbf{0} \in \partial f(\boldsymbol{x}) + \sum_{i=1}^n \mu_i^l \partial g_i^l(\boldsymbol{x}) + \sum_{i=1}^n \mu_i^u \partial g_i^u(\boldsymbol{x}).$$

A feasible point $\boldsymbol{x}$ satisfying the KKT optimality condition above is said to be a KKT point *associated with problem (1).*

Note that in the previous theorem we have $\partial g_i^l(\boldsymbol{x}) = \{\nabla g_i^l(\boldsymbol{x})\}$ and $\partial g_i^u(\boldsymbol{x}) = \{\nabla g_i^u(\boldsymbol{x})\}$ due to the differentiability of the constraints (see [5]).

Now, in addition to assuming that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous, the set $\mathcal{F} \cap \{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1)\,\}$ needs to be compact. Furthermore, we assume that each execution of the line search procedure is finite (i.e. that the modified semismoothness assumption (20) is valid).

We start the theoretical analysis of LMBM-B by studying the case when the algorithm terminates after a finite number of iterations: we prove that if Algorithm 2.1 stops at iteration k, then the point $\boldsymbol{x}_k$ is a KKT point for problem (1). Then, we prove that Algorithm 2.1 does not stop at a non-KKT-point and, finally, we prove that in the case of an infinite sequence $\{\boldsymbol{x}_k\} \subset \mathcal{F}$, every accumulation point $\bar{\boldsymbol{x}}$ of the sequence $\{\boldsymbol{x}_k\}$ generated by the algorithm is a KKT point for problem (1). In all of the above, we assume that the final accuracy tolerance ε is equal to zero.

Remark 3.1. The sequence $\{\boldsymbol{x}_k\}$ generated by Algorithm 2.1 is bounded by assumption and the monotonicity of the sequence $\{f_k\}$ which, in turn, is obtained due to condition (11) being satisfied for serious steps and the fact that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps. The sequence $\{\boldsymbol{y}_k\}$ is also bounded, since $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ for serious steps, $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| \le t_{max}C$ for null steps by (10), $t^k_{max} \le t_{max}$, 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. By the local boundedness and the upper semi-continuity of the subdifferential, we obtain the boundedness of subgradients $\boldsymbol{\xi}_k$ and their convex combinations (see [5]). Finally, the $\mathcal{P}_{\boldsymbol{x}}$-projections of subgradients are also bounded by definition.

Lemma 3.2. *Suppose that Algorithm 2.1 is not terminated before the kth iteration. Then there exist numbers $\lambda^{k,j} \ge 0$ for $j = 1, \ldots, k$ and $\tilde{\alpha}_k \ge 0$ such that*

$$(\tilde{\boldsymbol{\xi}}_k, \tilde{\alpha}_k) = \sum_{j=1}^{k} \lambda^{k,j}(\boldsymbol{\xi}_j, \|\boldsymbol{y}_j - \boldsymbol{x}_k\|), \qquad \sum_{j=1}^{k} \lambda^{k,j} = 1, \quad \textit{and} \quad \tilde{\beta}_k \ge \gamma \tilde{\alpha}_k^2.$$

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

Lemma 3.3. *Let $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ be given and suppose that there exist vectors $\bar{\boldsymbol{\zeta}}$, $\bar{\boldsymbol{\xi}}_j$, $\bar{\boldsymbol{y}}_j$, and numbers $\bar{\lambda}_j \ge 0$ for $j = 1, \ldots, l$, where $l \ge 1$, such that*

$$\begin{aligned}
&(\bar{\boldsymbol{\zeta}}, 0) = \sum_{j=1}^{l} \bar{\lambda}_j(\bar{\boldsymbol{\xi}}_j, \|\bar{\boldsymbol{y}}_j - \bar{\boldsymbol{x}}\|),\\
&\bar{\boldsymbol{\xi}}_j \in \partial f(\bar{\boldsymbol{y}}_j), \quad j = 1, \ldots, l,\\
&\sum_{j=1}^{l} \bar{\lambda}_j = 1.
\end{aligned}$$

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

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

Theorem 3.4. *If Algorithm 2.1 terminates in the kth iteration, then the point $\boldsymbol{x}_k$ is a KKT point for problem (1).*

Proof. Let us first point out that $\tilde{\beta}_k \ge 0$ for all k by (19), (16), and Step 1 in Algorithm 2.1. Due to (17), (18), and the positive definiteness of D_k, we have

$$w_k \ge 2\tilde{\beta}_k \qquad \text{and} \qquad w_k \ge \sigma \|\, \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k\,]\,\|^2. \tag{21}$$

Since Algorithm 2.1 terminates in Step 2, the fact that we have chosen $\varepsilon = 0$ implies that we have $w_k = 0$. Thus, $\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k\,] = \boldsymbol{0}$ and $\tilde{\beta}_k = \tilde{\alpha}_k = 0$ by (21) and Lemma 3.2.

Moreover, by Lemma 3.2 and by using Lemma 3.3 with

$$\begin{aligned} &\bar{\boldsymbol{x}} = \boldsymbol{x}_k, && l = k, && \bar{\boldsymbol{\zeta}} = \tilde{\boldsymbol{\xi}}_k, \\ &\bar{\boldsymbol{\xi}}_j = \boldsymbol{\xi}_j, && \bar{\boldsymbol{y}}_j = \boldsymbol{y}_j, && \bar{\lambda}_j = \lambda^{k,j} \quad \text{for } j \leq k, \end{aligned}$$

we obtain $\tilde{\boldsymbol{\xi}}_k \in \partial f(\boldsymbol{x}_k)$.

Let us first consider the case when none of the variables is on the boundary at point $\boldsymbol{x}_k$ (apropos, we always have $\boldsymbol{x}_k = \boldsymbol{x}_m$ for both serious and null steps). Then, $\tilde{\boldsymbol{\xi}}_k = \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]$ by (13). Thus, $\mathbf{0} = \tilde{\boldsymbol{\xi}}_k \in \partial f(\boldsymbol{x}_k)$ and $\boldsymbol{x}_k$ is (an unconstrained) KKT point for problem (1).

If some of the variables $x_{k,i}$, $i \in \{1, \ldots, n\}$ are on the boundary at point $\boldsymbol{x}_k$ we may have $\tilde{\xi}_{k,i} \neq 0$ for those variables (for the other variables we have $\tilde{\xi}_{k,i} = 0$ by (13)). Now, due to fact that $g_i^l(\boldsymbol{x}_k) = 0$ when $x_{k,i} = x_i^l$ and $g_i^u(\boldsymbol{x}_k) = 0$ when $x_{k,i} = x_i^u$, and by choosing $\mu_i^l, \mu_i^u = 0$ for free variables (subject to bounds at $\boldsymbol{x}_k$), we obtain $\mu_i^l g_i^l(\boldsymbol{x}_k) = 0$ and $\mu_i^u g_i^u(\boldsymbol{x}_k) = 0$ for all $i \in \{1, \ldots, n\}$.

Since Algorithm 2.1 stops in Step 2, we have $\tilde{\xi}_{k,i} \geq 0$ for all i such that $x_{k,i} = x_i^l$ and $\tilde{\xi}_{k,i} \leq 0$ for all i such that $x_{k,i} = x_i^u$. Now, by first noting that $\nabla g_i^l(\boldsymbol{x}_k)$ is a vector with ith component equal to -1 and all other components equal to zero, while $\nabla g_i^u(\boldsymbol{x}_k)$ has its ith component equal to 1 and all other components equal to zero, and then by choosing $\mu_i^l = \tilde{\xi}_{k,i}$, if $x_{k,i} = x_i^l$, and $\mu_i^u = -\tilde{\xi}_{k,i}$, if $x_{k,i} = x_i^u$, we obtain

$$\mathbf{0} = \tilde{\boldsymbol{\xi}}_k + \sum_{i=1}^{n} \mu_i^l \nabla g_i^l(\boldsymbol{x}_k) + \sum_{i=1}^{n} \mu_i^u \nabla g_i^u(\boldsymbol{x}_k)$$

with $\mu_i^l, \mu_i^u \geq 0$ for all $i \in \{1, \ldots, n\}$. Thus, by Theorem (3.1), the point $\boldsymbol{x}_k$ is a KKT point for problem (1). □

From now on, we suppose that Algorithm 2.1 does not terminate. We first study the case when $w_k = 0$ for some k but at least one component of the subgradient vector $\tilde{\boldsymbol{\xi}}_k$ is of the wrong sign.

Lemma 3.5. *Suppose that Algorithm 2.1 generates a point $\boldsymbol{x}_k \in \mathcal{F}$ such that $w_k = 0$ but either $\tilde{\xi}_{k,i} < 0$ for some i such that $x_{k,i} = x_i^l$, or $\tilde{\xi}_{k,i} > 0$ for some i such that $x_{k,i} = x_i^u$ (or both). Then, the next step is a serious step with $\boldsymbol{x}_{k+1} \neq \boldsymbol{x}_k$.*

Proof. For simplicity, we first consider the case when only one variable, say $x_{k,i}$, is on the boundary. From this starting point, the proof can be easily generalized.

Due to the fact that $w_k = 0$, we have $\mathcal{P}_{\boldsymbol{x}_k}[\tilde{\boldsymbol{\xi}}_k] = \mathbf{0}$ (i.e. we have $\tilde{\xi}_{k,j} = 0$ for all $j \neq i$) and, thus, we can restrict our consideration to a single component i. Let us first suppose that $x_{k,i} = x_i^u$. Since, in this case, $\xi_{k,i} > 0$, the generalized Cauchy point $x_{k,i}^c < x_{k,i} = x_i^u$ (see (3)). Thus, either i is not in $\mathcal{I}_A^k$ or $x_{k,i}^c$ is on the lower bound. In the former case, we simply obtain $\boldsymbol{d}_k^* = -D_k \tilde{\boldsymbol{\xi}}_k$. In the latter case, we have $\boldsymbol{b}_k = A_k^T(\boldsymbol{x}_k^c - \boldsymbol{x}_k) = x_i^l - x_i^u$. From (6) we obtain $d_i^* = x_i^l - x_i^u < 0$ and by (8) we have $d_j^* = -(D_k)_j \tilde{\boldsymbol{\xi}}_k$ for all $j \neq i$ ($(D_k)_j$ denotes the jth row of matrix D_k). In both cases it is clear that $(\boldsymbol{d}_k^*)^T \tilde{\boldsymbol{\xi}}_k < 0$.

Now, the search direction $\boldsymbol{d}_k$ is given by $\boldsymbol{d}_k = \boldsymbol{x}_k^c + \alpha_k^*(\boldsymbol{x}_k + \boldsymbol{d}_k^* - \boldsymbol{x}_k^c) - \boldsymbol{x}_k$ with $\alpha_k^* \in (0, 1]$ defined in (9). Since $\tilde{\xi}_{k,j} = 0$ for all $j \neq i$ and $\xi_{k,i} > 0$ we obtain

$$\boldsymbol{d}_k^T \tilde{\boldsymbol{\xi}}_k = ((1 - \alpha_k^*)(x_{k,i}^c - x_{k,i}) + \alpha_k^* d_i^*)\xi_{k,i} < 0.$$

That is, $\boldsymbol{d}_k$ is a descent direction for the objective function.

In the line search procedure, the initial step size $t_k^I \in [t_{min}, t_{max}^k]$ was chosen so as to minimize an approximation of $f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k)$ (for details see [34]). Now, we have $w_k = 0$, $\boldsymbol{d}_k$ is a descent direction, and we initialize $t = t_k^I$. Thus, for sufficiently small $t_{min} > 0$, we have

$$f(\boldsymbol{x}_k + t\theta_k \boldsymbol{d}_k) \le f(\boldsymbol{x}_k) - \varepsilon_L t w_k = f(\boldsymbol{x}_k) \qquad \text{and} \qquad t \ge t_{min} \tag{22}$$

in Step 2 of Algorithm 2.2 (in the first iteration of the algorithm). Therefore, a serious step with $\boldsymbol{x}_{k+1} \neq \boldsymbol{x}_k$ is taken.

Now suppose that, in addition to $x_{k,i}$, some other variable $x_{k,j}$ is on the boundary. Suppose also that, unlike for $x_{k,i}$, $\tilde{\xi}_{k,j}$ is of the right sign, let us say $x_{k,j} = x_j^u$ and $\tilde{\xi}_{k,j} \le 0$. The generalized Cauchy point is now given by $x_{k,j}^c = x_{k,j} = x_j^u$. That is, we have $j \in \mathcal{I}_A^k$ and $\boldsymbol{b}_k = A_k^T(\boldsymbol{x}_k^c - \boldsymbol{x}_k) = 0$ (for simplicity we assume that $i \notin \mathcal{I}_A^k$). By (6) we obtain $d_{k,j} = 0$ and the result above holds also in this case.

A similar procedure can be followed in the case when $x_{k,i} = x_i^l$ and/or $x_{k,j} = x_j^l$ and the proof can be easily generalized to the case where more variables are on the boundary. □

The previous lemma proves that Algorithm 2.1 does not stop at a non-KKT-point but moves away from the boundary when necessary. Now, we suppose that Algorithm 2.1 does not terminate and that $w_k > 0$ for all k.

Lemma 3.6. *Suppose that there exists a point $\bar{\boldsymbol{x}} \in \mathcal{F}$ 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 $\bar{\boldsymbol{x}}$ is a KKT point for problem (1).*

Proof. Let us denote by $\mathcal{J}$ the set $\{1, \ldots, n+2\}$. Using the fact that $\boldsymbol{\xi}_k \in \partial f(\boldsymbol{y}_k)$ for all $k \ge 1$, Lemma 3.2, and Carathéodory's theorem (see e.g. [15]), we deduce that there exist vectors $\boldsymbol{y}^{k,j}$ and $\boldsymbol{\xi}^{k,j}$, and numbers $\lambda^{k,j} \ge 0$ and $\tilde{\alpha}_k$ for $j \in \mathcal{J}$ and $k \ge 1$, such that

$$\begin{aligned}
&(\tilde{\boldsymbol{\xi}}_k, \tilde{\alpha}_k) = \sum_{j\in\mathcal{J}} \lambda^{k,j}(\boldsymbol{\xi}^{k,j}, \|\boldsymbol{y}^{k,j} - \boldsymbol{x}_k\|), \\
&\boldsymbol{\xi}^{k,j} \in \partial f(\boldsymbol{y}^{k,j}), \\
&\sum_{j\in\mathcal{J}} \lambda^{k,j} = 1,
\end{aligned} \tag{23}$$

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

From the boundedness of $\{\boldsymbol{y}_k\}$ (see Remark 3.1), we obtain the existence of points $\boldsymbol{y}_j^*$ $(j \in \mathcal{J})$, and an infinite set $\mathcal{K}_0 \subset \mathcal{K}$ satisfying $\{\boldsymbol{y}^{k,j}\}_{k\in\mathcal{K}_0} \to \boldsymbol{y}_j^*$ for $j \in \mathcal{J}$. The boundedness of $\{\boldsymbol{\xi}_k\}$ and $\{\lambda^{k,j}\}$ gives us the existence of vectors $\boldsymbol{\xi}_j^* \in \partial f(\boldsymbol{y}_j^*)$, numbers λ_j^* for $j \in \mathcal{J}$, and an infinite set $\mathcal{K}_1 \subset \mathcal{K}_0$ satisfying $\{\boldsymbol{\xi}^{k,j}\}_{k\in\mathcal{K}_1} \to \boldsymbol{\xi}_j^*$ and $\{\lambda^{k,j}\}_{k\in\mathcal{K}_1} \to \lambda_j^*$ for $j \in \mathcal{J}$.

It can be seen from (23) that

$$\lambda_j^* \ge 0 \quad \text{for } j \in \mathcal{J}, \qquad \text{and} \qquad \sum_{j\in\mathcal{J}} \lambda_j^* = 1.$$

From the fact that $\{w_k\}_{k\in\mathcal{K}} \to 0$, equation (21), and Lemma 3.2, we obtain

$$\{\tilde{\beta}_k\}_{k\in\mathcal{K}} \to 0 \qquad \text{and} \qquad \{\tilde{\alpha}_k\}_{k\in\mathcal{K}} \to 0.$$

By letting $k \in \mathcal{K}_1$ approach infinity in (23), and by using Lemma 3.3 with

$$\begin{aligned} &\bar{\boldsymbol{\zeta}} = \tilde{\boldsymbol{\xi}}_k, && \bar{\boldsymbol{\xi}}_j = \boldsymbol{\xi}_j^*, && \bar{\boldsymbol{y}}_j = \boldsymbol{y}_j^*, \\ &l = n+2, && \bar{\lambda}_j = \lambda_j^* && \text{for } j \leq l, \end{aligned}$$

we obtain $\tilde{\boldsymbol{\xi}}_k \in \partial f(\bar{\boldsymbol{x}})$.

If none of the variables is on the boundary[3] at $\bar{\boldsymbol{x}}$, we have $\tilde{\boldsymbol{\xi}}_k = \mathcal{P}_{\bar{\boldsymbol{x}}}[\tilde{\boldsymbol{\xi}}_k\,]$ by (13). Since $\{w_k\}_{k \in \mathcal{K}} \to 0$, we have $\{\mathcal{P}_{\bar{\boldsymbol{x}}}[\tilde{\boldsymbol{\xi}}_k\,]\}_{k \in \mathcal{K}} \to \mathbf{0}$ by (21). Thus, $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$ and $\bar{\boldsymbol{x}}$ is (an unconstrained) KKT point for problem (1).

Suppose now that some of the variables are on the boundary at $\bar{\boldsymbol{x}}$. By Lemma 3.5 a serious step with $\boldsymbol{x}_{k+1} \neq \boldsymbol{x}_k$ occurs (i.e. $\boldsymbol{x}_k$ is not an accumulation point) if $w_k = 0$ but some components of the aggregate subgradient are of the wrong sign. Thus, we can restrict our consideration to the case when the components of $\tilde{\boldsymbol{\xi}}_k$ are either zero or of the right/correct sign. Since $\{w_k\}_{k \in \mathcal{K}} \to 0$, we have $\{\mathcal{P}_{\bar{\boldsymbol{x}}}[\tilde{\boldsymbol{\xi}}_k\,]\}_{k \in \mathcal{K}} \to \mathbf{0}$ by (21). Now, as in the proof of Theorem 3.4, we can choose $\mu_i^l, \mu_i^u \geq 0$ such that $\mu_i^l g_i^l(\bar{\boldsymbol{x}}) = 0$ and $\mu_i^u g_i^u(\bar{\boldsymbol{x}}) = 0$ for all $i \in \{1, \ldots, n\}$ and

$$\mathbf{0} = \tilde{\boldsymbol{\xi}}_k + \sum_{i=1}^n \mu_i^l \nabla g_i^l(\bar{\boldsymbol{x}}) + \sum_{i=1}^n \mu_i^u \nabla g_i^u(\bar{\boldsymbol{x}}).$$

Thus, by Theorem (3.1) the point $\bar{\boldsymbol{x}}$ is a KKT point for problem (1). □

Lemma 3.7. *Suppose that the number of serious steps in Algorithm 2.1 is finite and the last serious step occurred at iteration $m-1$. Then there exists a number $k^* \geq m$, such that*

$$\begin{gathered} \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,]^T D_{k+1} \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,] \leq \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,] \qquad \textit{and} \\ \operatorname{tr}(D_k) < \frac{3}{2} n \end{gathered}$$

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

Proof. The result is due to the safeguarded SR1 update used (see [12, 14]). The proof is similar to the proof of Lemma 7 in [14]. □

Lemma 3.8. *Suppose that the number of serious steps is finite and the last serious step occurred at iteration $m-1$. Then, $\boldsymbol{x}_m$ is a KKT point for problem (1).*

Proof. From (14), (15), (16), (17), (18), and Lemma 3.7 we obtain

$$\begin{aligned} w_{k+1} &= \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,]^T D_{k+1} \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,] + 2\tilde{\beta}_{k+1} \\ &\leq \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,] + 2\tilde{\beta}_{k+1} \\ &\leq \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k\,]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k\,] + 2\tilde{\beta}_k = w_k \end{aligned} \tag{24}$$

for $k \geq k^*$ with k^* defined as in Lemma 3.7 (for simplicity, we write D_k instead of $D_k + \sigma I$ if $i_{CN} = 1$, so that (17) and (18) coincide). The last inequality in (24) follows from the fact that the pair $(\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_{k+1}\,], \tilde{\beta}_{k+1})$ minimizes function (14) over all

[3] If none of the variables are on the boundary, this proof is in fact exactly the same as the proof of Lemma 3.4 in [34].

convex combinations of pairs $(\mathcal{P}_{\boldsymbol{x}_m}[\boldsymbol{\xi}_m], \beta_m)$, $(\mathcal{P}_{\boldsymbol{x}_m}[\boldsymbol{\xi}_{k+1}], \beta_{k+1})$ and $(\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k], \tilde{\beta}_k)$. In addition, the line search procedure guarantees that we have

$$-\beta_{k+1} - \mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_m}[\boldsymbol{\xi}_{k+1}] \geq -\varepsilon_R w_k$$

for all $k \geq m$. Now, due to the boundedness of $\mathcal{P}_{\boldsymbol{x}_m}[\boldsymbol{\xi}_{k+1}]$, $\mathcal{P}_{\boldsymbol{x}_m}[\tilde{\boldsymbol{\xi}}_k]$, and D_k (see Remark 3.1 and Lemma 3.7) it can be proved that $w_k \to 0$ (the proof is rather similar to part(ii) of the proof of Lemma 3.6 in [34]). Now, since $\boldsymbol{x}_k = \boldsymbol{x}_m$ for all $k \geq m$, we have that $\boldsymbol{x}_k \to \boldsymbol{x}_m$. Therefore, by Lemma 3.6, $\boldsymbol{x}_m$ is a KKT point for problem (1). □

Remark 3.2. In the proof of Lemma 3.8, we alluded to the fact that in the case of consecutive null steps, $w_k \to 0$. On the other hand, by Lemma 3.5, a serious step occurs if $w_k = 0$ but some components of the aggregate subgradient vector are of the wrong sign. Thus, if $w_k \to 0$ we either have a KKT point for the problem or a serious step with $\boldsymbol{x}_{k+1} \neq \boldsymbol{x}_k$ occurs.

Theorem 3.9. *Every accumulation point $\bar{\boldsymbol{x}}$ of the sequence $\{\boldsymbol{x}_k\}$ generated by Algorithm 2.1 is a KKT point for problem (1).*

Proof. The proof is similar to the proof of Theorem 9 in [14], except that here we use Lemma 3.6 instead of Lemma 6 of [14]. □

4 Numerical Experiments

We now compare numerically the proposed globally convergent limited memory bundle method `LMBM-B` to the proximal bundle method `PBNCGC` (version 2.0, [25, 26]) for some nonsmooth large-scale test problems. We use the solver `PBNCGC` as a benchmark since the proximal bundle method is the most frequently used bundle method in nonsmooth optimization. In addition, we compare the new version of `LMBM-B` to the older, non-globally convergent, version `LMBM-B-OLD` [19]. The experiments were performed on an Intel® Core™ 2 CPU 1.80GHz and all the algorithms were implemented in Fortran77 with double-precision arithmetic.

The solvers were tested using 10 nonsmooth academic minimization problems described in [13][4]. The number of variables used in our experiments were 1000, 2000, and 4000. The problems in [13] are unconstrained but we imposed the additional bounds

$$x_i^* + 0.1 \leq x_i \leq x_i^* + 1.1 \quad \text{for all odd } i,$$

where $\boldsymbol{x}^*$ denotes the solution for the unconstrained problem. If the original starting point given in [13] was not feasible, we simply projected it to the feasible region.

The solvers were tested with a relatively small amount of stored subgradient information. That is, the size of the bundle m_ξ was set to 10 for `LMBM-B` and `LMBM-B-OLD` and to 100 for `PBNCGC` (since previous experiments [12, 14] have shown that a larger bundle usually works better with `PBNCGC`). For convex problems (problems 1 – 5 in [13]), we used the distance measure parameter $\gamma = 0$ and for nonconvex problems (problems 6 – 10 in [13]), we used the value $\gamma = 0.5$ with all of the solvers. The final accuracy parameter $\varepsilon = 10^{-5}$ was used in all cases. For all other parameters, we used the default settings of the solvers.

[4] All of these problems can be downloaded from the website http://napsu.karmitsa.fi/lmbm.

In addition to the usual stopping criteria of the solvers, we terminated the experiments if the elapsed CPU time exceeded half an hour.

We say that a solver finds the solution with respect to a tolerance $\varepsilon_s > 0$ if

$$\frac{f_{best} - f_{opt}}{1 + |f_{opt}|} \leq \varepsilon_s,$$

where f_{best} is a solution obtained with the solver and f_{opt} is the best known (or optimal) solution. In our experiments, we used a tolerance of $\varepsilon_s = 10^{-3}$. Otherwise, we considered the result a failure.

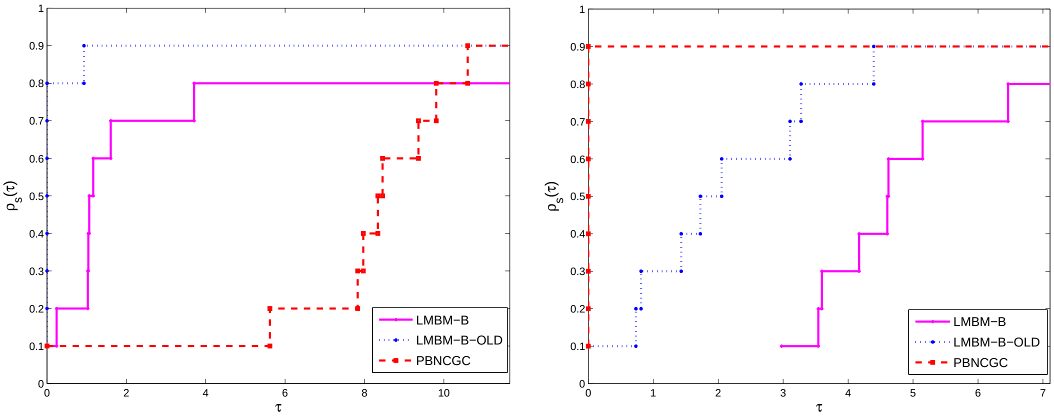
The results are summarized in Figures 2, 3, and 4. The results are analyzed using the performance profiles introduced in [9]. As performance measures, we use computation times (in Figures 2–4(a)) and numbers of function evaluations (in Figures 2–4(b)). In the performance profiles, the value of $\rho_s(\tau)$ at $\tau = 0$ gives the percentage of test problems for which the corresponding solver is the best and the value of $\rho_s(\tau)$ at the rightmost abscissa gives the percentage of test problems that the corresponding solver can solve (this does not depend on the measured performance). Moreover, the relative efficiency of each solver can be directly seen from the performance profiles: the higher the particular curve, the better the corresponding solver. For more information on performance profiles, see [9].

In Figures 2(a), 3(a), and 4(a), we see the superiority of the different variants of LMBM-B when comparing the computational times; the computation times elapsed with `LMBM-B` and `LMBM-B-OLD` were typically hundreds of times shorter than those of `PBNCGC`. On the other hand, there was not a very big difference in the computational times between the different variants of LMBM-B, although the globally convergent version `LMBM-B` usually needed more computation time than the older one. This is due to the fact that fewer function evaluations are required with `LMBM-B-OLD` (see Figures 2(b), 3(b), and 4(b)). The increased number of function evaluations needed with `LMBM-B` is probably due to a less accurate search direction caused by the stark projection of subgradients. Thus, different projection possibilities need to be studied.

In each case the proximal bundle solver `PBNCGC` needed less function evaluations than the different variants of LMBM-B. However, as can be seen when comparing the computational times, each individual iteration with `PBNCGC` was much more costly than that with `LMBM-B` or `LMBM-B-OLD`. Indeed, with `PBNCGC`, all the but two of the problems with 4000 variables were terminated due to the time limit being exceeded (problems 2 and 6 were the exceptions). This also explains the large number of failures 50% (inaccurate results) obtained with `PBNCGC` (see Figure 4).

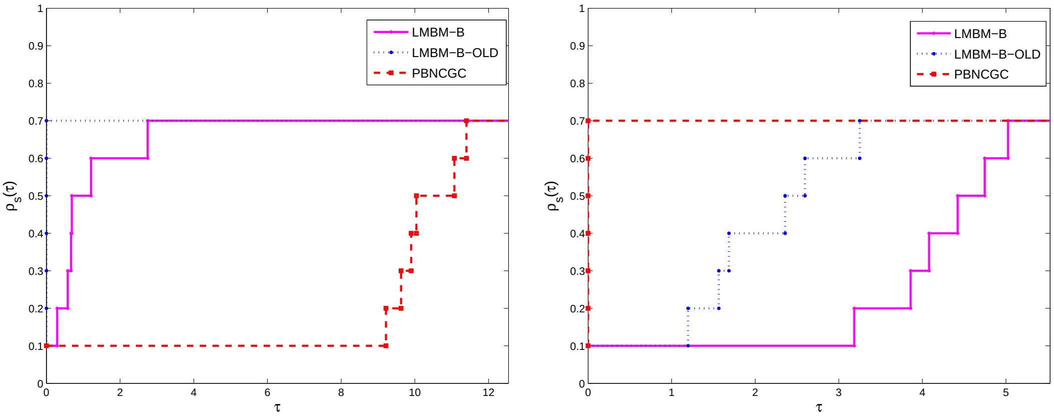
The new variant `LMBM-B` failed to solve two of the problems (problems 1 and 2) with any number of variables tested (the older variant `LMBM-B-OLD` succeed in solving problem 1 with 1000 variables and failed otherwise). These failures were quite predictable, since both of these problems are reported to be difficult to solve using limited memory bundle method even without the bound constraints [13]. With 4000 variables the new variant `LMBM-B` succeed in solving more problems than the other solvers.

To sum up, both variants of LMBM-B were clearly faster than the proximal bundle solver `PBNCGC` and had the same magnitude of required computational time. The enhanced variant `LMBM-B` has theoretical convergence properties while being only slightly slower than its predecessor `LMBM-B-OLD`.



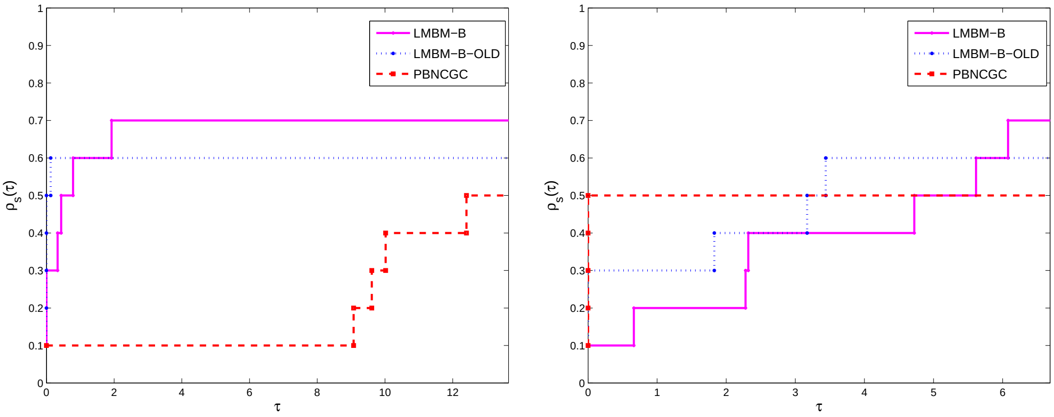
(a) CPU time (b) Function evaluations

Figure 2: Results for bound constrained problems with 1000 variables.



(a) CPU time (b) Function evaluations

Figure 3: Results for bound constrained problems with 2000 variables.



(a) CPU time (b) Function evaluations

Figure 4: Results for bound constrained problems with 4000 variables.

5 Conclusions

In this paper, we have described a new variant of the limited memory bundle method LMBM-B for bound constrained nonsmooth large-scale optimization. The new variant has most of the efficiency of its predecessor [19] with the enhancement of having also the theoretical convergence properties. Namely, through some projection and correction operations together with a modified line search we have managed to prove the global convergence of the method for locally Lipschitz continuous functions, which are not necessarily differentiable or convex.

The main advantages of LMBM-B are its low computational cost per iteration, the relatively low storage requirements, and its ability to solve problems of unknown structure and problems for which the differentiability or convexity cannot be confirmed. The fact that LMBM-B only generates feasible points may be essential in the case when the objective function or the subgradient values are undefined or difficult to compute when some of the constraints are violated. Furthermore, this feature can be an advantage in many industrial applications, where function evaluation may be very expensive. Since any intermediate solution can be employed, the iterations can be stopped whenever the result is satisfactory. Due to this feasibility, the efficiency of the method, and the fact that the objective function need not to be differentiable or convex, we expect LMBM-B to be very useful in solving optimization problems arising in real world modeling.

Acknowledgements

This work was financially supported by the University of Jyväskylä (Finland) and the University of Turku (Finland).

Appendix

The limited memory variable metric matrices used in our algorithm are represented in the compact matrix form originally described in [4].

Let us denote by $\hat{m}_c$ the user-specified maximum number of stored correction pairs ($3 \le \hat{m}_c$) 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

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

where the correction pairs $(\boldsymbol{s}_i, \boldsymbol{u}_i)$, $(i < k)$ are obtained in Step 5 of Algorithm 2.1.

The inverse limited memory BFGS update is defined by the formula

$$D_k = \vartheta_k I + \begin{bmatrix} S_k & \vartheta_k U_k \end{bmatrix} \begin{bmatrix} (R_k^{-1})^T (C_k + \vartheta_k U_k^T U_k) R_k^{-1} & -(R_k^{-1})^T \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^T \\ \vartheta_k U_k^T \end{bmatrix}, \tag{25}$$

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

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

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

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

and ϑ_k is a positive scaling parameter.

A similar representation of the direct limited memory BFGS update can be written as

$$B_k = \frac{1}{\vartheta_k} I - \begin{bmatrix} \frac{1}{\vartheta_k} S_k & U_k \end{bmatrix} \begin{bmatrix} \frac{1}{\vartheta} S_k^T S_k & L_k \\ L_k^T & -C_k \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{\vartheta_k} S_k^T \\ U_k^T \end{bmatrix}, \tag{26}$$

where

$$L_k = S_k^T U_k - R_k.$$

Note that we have $\hat{m}_1 = 0$, $\vartheta_1 = 1$ and, thus, at the first iteration $B_1 = D_1 = I$.

The inverse limited memory SR1 update is defined by

$$D_k = \vartheta_k I - (\vartheta_k U_k - S_k)(\vartheta_k U_k^T U_k - R_k - R_k^T + C_k)^{-1}(\vartheta_k U_k - S_k)^T \tag{27}$$

and, correspondingly, the direct SR1 update is defined by

$$B_k = \frac{1}{\vartheta_k} I + (U_k - \frac{1}{\vartheta_k} S_k)(L_k + L_k^T + C_k - \frac{1}{\vartheta_k} S_k^T S_k)^{-1}(U_k - \frac{1}{\vartheta_k} S_k)^T. \tag{28}$$

With SR1 updates we use the value $\vartheta_k = 1$ for all k.

Lemma 5.1. *The condition*

$$-\boldsymbol{s}_i^T \boldsymbol{u}_i + (t^i \theta_i)^2 \max\{(\boldsymbol{x}_i^c - \boldsymbol{x}_i)^T B_i (\boldsymbol{x}_i^c - \boldsymbol{x}_i), (-A_i \boldsymbol{\mu}_i^* - \tilde{\boldsymbol{\xi}}_i)^T \boldsymbol{d}_i^*\} < 0 \tag{29}$$

for all $i = 1, \dots, k-1$ assures the positive definiteness of matrices obtained by limited memory SR1 updates.

Proof. The limited memory SR1 updates D_k and B_k ($k \geq 1$, $D_1 = B_1 = I$) are positive definite if

$$\boldsymbol{s}_i^T(\boldsymbol{u}_i - B_i \boldsymbol{s}_i) > 0$$

for all $i = 1, \dots, k$ (see the proof of Lemma 10 in [14]). Now, since $\boldsymbol{d}_i = (1-\alpha_i^*)(\boldsymbol{x}_i^c - \boldsymbol{x}_i) + \alpha_i^* \boldsymbol{d}_i^*$, $B_i \boldsymbol{d}_i^* = -A_i \boldsymbol{\mu}_i^* - \tilde{\boldsymbol{\xi}}_i$, and due to the convexity of the function $\|\cdot\|^2$, we obtain

$$\begin{aligned}
\boldsymbol{s}_i^T B_i \boldsymbol{s}_i &= (t^i \theta_i)^2 \boldsymbol{d}_i^T B_i \boldsymbol{d}_i \\
&= (t^i \theta_i)^2 ((1-\alpha_i^*)(\boldsymbol{x}_i^c - \boldsymbol{x}_i) + \alpha_i^* \boldsymbol{d}_i^*)^T B_i ((1-\alpha_i^*)(\boldsymbol{x}_i^c - \boldsymbol{x}_i) + \alpha_i^* \boldsymbol{d}_i^*) \\
&= (t^i \theta_i)^2 \|(1-\alpha_i^*) N_i (\boldsymbol{x}_i^c - \boldsymbol{x}_i) + \alpha_i^* N_i \boldsymbol{d}_i^*\|^2 \\
&\leq (t^i \theta_i)^2 \max\{\|N_i(\boldsymbol{x}_i^c - \boldsymbol{x}_i)\|^2, \|N_i \boldsymbol{d}_i^*\|^2\} \\
&= (t^i \theta_i)^2 \max\{(\boldsymbol{x}_i^c - \boldsymbol{x}_i)^T B_i (\boldsymbol{x}_i^c - \boldsymbol{x}_i), (\boldsymbol{d}_i^*)^T B_i \boldsymbol{d}_i^*\} \\
&= (t^i \theta_i)^2 \max\{(\boldsymbol{x}_i^c - \boldsymbol{x}_i)^T B_i (\boldsymbol{x}_i^c - \boldsymbol{x}_i), (-A_i \boldsymbol{\mu}_i^* - \tilde{\boldsymbol{\xi}}_i)^T \boldsymbol{d}_i^*\},
\end{aligned}$$

where we have denoted $B_i = N_i^T N_i$. Therefore, if condition (29) is valid, we have $\boldsymbol{s}_i^T(\boldsymbol{u}_i - B_i \boldsymbol{s}_i) > 0$ for all $i = 1, \dots, k-1$. □

Condition (29) also guarantees $\boldsymbol{s}_i^T \boldsymbol{u}_i > 0$ for all $i = 1, \dots, k-1$ which is the classical condition for the positive definiteness of the (limited memory) BFGS updates. Since we check condition (29) also during the limited memory BFGS update before updating matrices, all the matrices D_k and B_k formed in LMBM-B are positive definite.

Note that testing the positive definiteness does not require any additional matrix calculations since $(\boldsymbol{x}_i^c - \boldsymbol{x}_i)^T B_i (\boldsymbol{x}_i^c - \boldsymbol{x}_i)$ has already been calculated in previous iteration during the computation of the generalized Cauchy point. If there are no bounds on the variables, this positive definiteness condition reverts to the condition of the unconstrained version (see [14]).

In order to guarantee the global convergence of LMBM-B, the boundedness of matrices $B_i = D_i^{-1}$ is required (we say that a matrix is bounded if its eigenvalues lie in a compact interval that does not contain zero). The utilization of correction in Step 2 of Algorithm 2.1 is equivalent to adding a positive definite matrix σI to matrix D_k. Since the limited memory representation of matrix D_k (or B_k) does not contain information about the correction σI that may have been added, we have to add it explicitly in Steps 3, 4, and 6 if $i_{CN} = 1$. In the case of inverse updates (25) and (27) (Steps 4 and 6 in Algorithm 2.1), this is obviously an easy task that does not require much additional computation. However, in Step 3 we have to calculate $B_k = (D_k + \sigma I)^{-1}$ instead of using formula (26) or (28). By using the Sherman-Morrison-Woodbury formula for $(D_k + \sigma I)^{-1}$, omitting k and denoting $\rho = \vartheta/(\vartheta + \sigma)$, the limited memory BFGS type of update can be written in the form

$$B = \frac{\rho}{\vartheta} I - \rho^2 \left[\tfrac{1}{\vartheta} S \quad U\right] \begin{bmatrix} \frac{\rho}{\vartheta} S^T S & \rho L - (1-\rho) R \\ \rho L^T - (1-\rho) R^T & -C - (1-\rho)\vartheta U^T U \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{\vartheta} S^T \\ U^T \end{bmatrix}, \tag{30}$$

and the limited memory SR1 type of update is given by

$$B = \tfrac{\rho}{\vartheta} I + \rho^2 (U - \tfrac{1}{\vartheta} S)((1-\rho)(U^T U - R - R^T + C) + \rho(L + L^T + C - \tfrac{1}{\vartheta} S^T S))^{-1} (U - \tfrac{1}{\vartheta} S)^T. \tag{31}$$

The middle matrix in (30) is indefinite. However, using a procedure rather similar to that given in [4] for (26), its inversion can be carried out using Cholesky factorization of the related matrix. Thus, to implement (30), only the Cholesky factorizations of two $\hat{m}_k \times \hat{m}_k$ symmetric positive definite matrices need to be computed (see [18]). Naturally, formulae (30) and (31) require more computations than the traditional formulae (26) and (28), but in these cases the calculations can also be done within $O(n)$ operations.

Finally, we remark that the basic assumption for bundle methods to converge that after a null step we have $\boldsymbol{z}^T D_{k+1} \boldsymbol{z} \leq \boldsymbol{z}^T D_k \boldsymbol{z}$ for all $\boldsymbol{z} \in \mathbb{R}^n$, is guaranteed by the special limited memory SR1 update [12, 14].

References

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

[2] Bihain, A. Optimization of upper semidifferentiable functions. *Journal of Optimization Theory and Applications 4* (1984), 545–568.

[3] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing 16*, 5 (1995), 1190–1208.

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

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

[6] Clarke, F. H., Ledyaev, Y. S., Stern, R. J., and Wolenski, P. R. *Nonsmooth Analysis and Control Theory*. Springer, New York, 1998.

[7] Conn, A. R., Gould, N. I. M., and Toint, P. L. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis 25*, 2 (1988), 433–460.

[8] Conn, A. R., Gould, N. I. M., and Toint, P. L. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation 50*, 182 (1988), 399–430.

[9] Dolan, E. D., and Moré, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming 91* (2002), 201–213.

[10] Fletcher, R. *Practical Methods of Optimization*, 2nd ed. John Wiley and Sons, Chichester, 1987.

[11] Gilbert, J.-C., and Lemaréchal, C. Some numerical experiments with variable-storage quasi-Newton algorithms. *Mathematical Programming 45* (1989), 407–435.

[12] Haarala, M. *Large-Scale Nonsmooth Optimization: Variable Metric Bundle Method with Limited Memory*. PhD thesis, University of Jyväskylä, Department of Mathematical Information Technology, 2004.

[13] Haarala, M., Miettinen, K., and Mäkelä, M. M. New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods and Software 19*, 6 (2004), 673–692.

[14] Haarala, N., Miettinen, K., and Mäkelä, M. M. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming 109*, 1 (2007), 181–205.

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

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

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

[18] Karmitsa, N., and Mäkelä, M. M. Globally convergent limited memory bundle algorithm for nondifferentiable programming subject to box constraints. TUCS Technical Report, No. 882, Turku Centre for Computer Science, Turku, 2008.

[19] Karmitsa, N., and Mäkelä, M. M. Adaptive limited memory bundle method for bound constrained large-scale nonsmooth optimization. to appear in *Optimization*, 2009.

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

[21] Lemaréchal, C., Strodiot, J.-J., and Bihain, A. On a bundle algorithm for nonsmooth optimization. In *Nonlinear Programming*, O. L. Mangasarian,

R. R. Mayer, and S. M. Robinson, Eds. Academic Press, New York, 1981, pp. 285–281.

[22] Liu, D. C., and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming 45* (1989), 503–528.

[23] Lukšan, L., and Vlček, J. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications 102*, 3 (1999), 593–613.

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

[25] Mäkelä, M. M. Multiobjective proximal bundle method for nonconvex nonsmooth optimization: Fortran subroutine MPBNGC 2.0. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, B. 13/2003 University of Jyväskylä, Jyväskylä, 2003.

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

[27] Mifflin, R. A modification and an extension of Lemaréchal's algorithm for nonsmooth minimization. *Matematical Programming Study 17* (1982), 77–90.

[28] Mistakidis, E. S., and 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.

[29] Moreau, J. J., Panagiotopoulos, P. D., and Strang, G., Eds. *Topics in Nonsmooth Mechanics*. Birkhäuser Verlag, Basel, 1988.

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

[31] Outrata, J., Kočvara, M., and Zowe, J. *Nonsmooth Approach to Optimization Problems With Equilibrium Constraints. Theory, Applications and Numerical Results*. Kluwert Academic Publisher, Dordrecht, 1998.

[32] Panier, E. R. An active set method for solving linearly constrained nonsmooth optimization problems. *Mathematical Programming 37* (1987), 269–292.

[33] Schramm, H., and Zowe, J. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization 2*, 1 (1992), 121–152.

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