Diagonal Bundle Method for Nonsmooth Sparse Optimization

N. Karmitsa[1]

Abstract: We propose an efficient diagonal bundle method D-Bundle for sparse nonsmooth, possibly nonconvex optimization. The convergence of the proposed method is proved for locally Lipschitz continuous functions that are not necessary differentiable nor convex. The numerical experiments have been made using problems with up to million variables. The results to be presented confirm the usability of the D-Bundle especially for extremely large-scale problems.

Keywords: Nondifferentiable optimization, sparse problems, bundle methods, diagonal variable metric methods.

1 Introduction

We introduce a new diagonal bundle method (D-Bundle) for solving unconstrained nonsmooth optimization (NSO) problems of the form

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

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is supposed to be locally Lipschitz continuous (LLC). Note that no differentiability or convexity assumptions for problem (P) are made. In particular, our aim is to design a method for solving problem (P) with large numbers of variables (i.e. problems with more than 1000 variables), where the Hessian of the problem — if it exists — is sparse.

NSO problems are encountered in many application areas: for instance, in economics [50], mechanics [47], engineering [46], control theory [17], optimal shape design [29], machine learning [32], and data mining [4, 12] including cluster analysis [18] and classification [2, 3, 10, 15]. Most of these problems are large-scale.

NSO problems are in general difficult to solve even when the size of the problem is small. The direct application of gradient-based methods to nonsmooth problems may lead to a failure in convergence, in optimality conditions, or in gradient approximation (see, e.g. [37]). Methods for solving NSO problems include subgradient methods (see e.g. [8, 9, 54, 56]), bundle methods (see

[1] Department of Mathematics and Statistics, University of Turku, FI-20014 Turku, Finland. E-mail: napsu@karmitsa.fi

e.g. [24, 31, 36, 40, 41, 43, 52, 53]), algorithms based on smoothing techniques [51], and the gradient sampling methods [13]. Most of these methods are developed for small-scale and/or convex problems and if applied to large-scale and/or nonconvex problems they may suffer from inefficiency and a lack of robustness.

Indeed, besides problematics of nonsmoothness and size of the problem, nonconvexity adds another challenge; NSO is traditionally based on convex analysis and most solution methods rely strongly on the convexity of the problem. Fortunately, several nonconvex algorithms have been introduced only recently, for instance, in [1, 13, 23, 22, 28, 35, 49]. Nevertheless, also most of these algorithms are developed only for small-scale problems.

In [25, 26, 27] a limited memory bundle method (LMBM) for large-scale nonconvex nonsmooth minimization was introduced and its convergence was proved for LLC functions. However, to compute the search direction LMBM uses a dense approximation to the variable metric matrix and, thus, it fails to solve some sparse problems (see, e.g. [7, 33]).

In this paper we introduce a diagonal bundle method D-Bundle for sparse nonconvex NSO. The idea of the new method is to combine LMBM with sparse matrix updating. Although it would be possible to use real sparsity pattern of the Hessian, we use here the diagonal update formula introduced in [30], since for this formula it is easy to check the positive definiteness of generated matrices. The convergence of the D-Bundle is proved for LLC functions. In addition, the numerical experiments has been made using NSO problems with up to million variables.

The paper is organized as follows. In Section 2 we recall some basic definitions and results from nonsmooth analysis. Furthermore, we discuss briefly the basic ideas of LMBM and sparse matrix updating. In Section 3 we introduce the new method and study its convergence properties. The results of the numerical experiments are presented and discussed in Section 4 and, finally, Section 5 concludes the paper.

2 Background

In this section, we first recall some basic definitions and results from nonsmooth analysis. Then, we discuss basic ideas of LMBM and, finally, we briefly describe the ideas of sparse matrix updating.

2.1 Preliminaries

We denote by $\|\cdot\|$ the Euclidean norm in $\mathbb{R}^n$ and by $\boldsymbol{a}^T\boldsymbol{b}$ the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ (bolded symbols are used for vectors). In addition, we denote by

$\|A\|_F$ the Frobenius norm of matrix $A \in \mathbb{R}^{n\times n}$. That is, we define

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}^2}.$$

The *subdifferential* $\partial f(\boldsymbol{x})$ [16] of a LLC function $f : \mathbb{R}^n \to \mathbb{R}$ at any point $\boldsymbol{x} \in \mathbb{R}^n$ is given by

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

where "conv" denotes the convex hull of a set. A vector $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is called a *subgradient*.

The point $\boldsymbol{x}^* \in \mathbb{R}^n$ is called *stationary* if $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. Stationarity is a necessary condition for local optimality and, in the convex case, it is also sufficient for global optimality. An optimization method is said to be *globally convergent* if starting from any arbitrary point $\boldsymbol{x}_1$ it generates a sequence $\{\boldsymbol{x}_k\}$ that converges to a stationary point $\boldsymbol{x}^*$, that is, $\{\boldsymbol{x}_k\} \to \boldsymbol{x}^*$ whenever $k \to \infty$.

2.2 Limited Memory Bundle Method

In this subsection, we describe the limited memory bundle algorithm (LMBM) by Karmitsa (née Haarala) et al. [25, 26, 27, 34] for solving general, possibly nonconvex, large-scale NSO problems. We assume that at every point $\boldsymbol{x}$ we can evaluate the value of the objective function $f(\boldsymbol{x})$ and one arbitrary subgradient $\boldsymbol{\xi}$ from the subdifferential $\partial f(\boldsymbol{x})$.

LMBM is a hybrid of the variable metric bundle methods [40, 57] and the limited memory variable metric methods (see e.g. [14]), where the first ones have been developed for small- and medium-scale nonsmooth optimization and the latter ones for smooth large-scale optimization. LMBM exploits the ideas of the variable metric bundle methods, namely the utilization of null steps, simple aggregation of subgradients, and the subgradient locality measures, but the search direction $\boldsymbol{d}_k$ is calculated using the limited memory approach. That is,

$$\boldsymbol{d}_k = -D^k \tilde{\boldsymbol{\xi}}_k,$$

where $\tilde{\boldsymbol{\xi}}_k$ is (an aggregate) subgradient and D^k is the limited memory variable metric update that, in the smooth case, represents the approximation of the inverse of the Hessian matrix. Note that the matrix D^k is not formed explicitly but the search direction $\boldsymbol{d}_k$ is calculated using the limited memory approach.

In order to determine a new step into the search direction $\boldsymbol{d}_k$, LMBM uses the so-called *line search procedure* (see [27, 57]): a new iteration point $\boldsymbol{x}_{k+1}$ and a new auxiliary point $\boldsymbol{y}_{k+1}$ are produced such that

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

with $\boldsymbol{y}_1 = \boldsymbol{x}_1$, where $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$ are step sizes, and $t_{max} > 1$ is the upper bound for the step size. A necessary condition for a *serious step* to be taken is to have

$$t_R^k = t_L^k > 0 \qquad \text{and} \qquad f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L^k t_R^k w_k, \tag{2}$$

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

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

$$t_R^k > t_L^k = 0 \qquad \text{and} \qquad -\beta_{k+1} + \boldsymbol{d}_k^T \boldsymbol{\xi}_{k+1} \geq -\varepsilon_R^k w_k, \tag{3}$$

where $\varepsilon_R^k \in (\varepsilon_L^k, 1/2)$ is a line search parameter and $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$. Moreover, β_{k+1} is the subgradient locality measure [38, 45] similar to standard bundle methods, that is,

$$\beta_{k+1} = \max\{|f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + (\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)^T \boldsymbol{\xi}_{k+1})|, \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|^2\}. \tag{4}$$

Here $\gamma \geq 0$ is a distance measure parameter supplied by the user. Parameter γ can be set to zero when f is convex.

In the case of a null step, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ but information about the objective function is increased because we store the auxiliary point $\boldsymbol{y}_{k+1}$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$.

Note that under some semismootheness assumptions the line search procedure used is guaranteed to find the step sizes t_L^k and t_R^k such that exactly one of the two possibilities — a serious step or a null step — occurs (see [57]).

LMBM uses the original subgradient $\boldsymbol{\xi}_k$ 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* 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 a simple quadratic function

$$\begin{aligned} \varphi(\lambda_1, \lambda_2, \lambda_3) = [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k]^T D^k [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k] \\ + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k). \end{aligned} \tag{5}$$

Here $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$ is the current subgradient (m denotes the index of the iteration after the latest serious step, i.e. $\boldsymbol{x}_k = \boldsymbol{x}_m$), $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ 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 subgradient locality measure and $\tilde{\beta}_k$ is the current aggregate subgradient locality measure ($\tilde{\beta}_1 = 0$). The optimal values λ_i^k, $i \in \{1, 2, 3\}$can be calculated by using simple formulae.

The resulting aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k+1}$ and aggregate subgradient locality measure $\tilde{\beta}_{k+1}$ are computed by the formulae

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad \text{and} \qquad \tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{6}$$

Due to this simple aggregation procedure only one trial point $\boldsymbol{y}_{k+1}$ and the corresponding subgradient $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ need to be stored.

The aggregation procedure gives us a possibility to retain the global convergence without solving the quite complicated quadratic direction finding problem (see e.g. [7]) appearing in standard bundle methods. Note that the aggregate values are computed only if the last step was a null step. Otherwise, we set $\tilde{\boldsymbol{\xi}}_{k+1} \boldsymbol{\xi}_{k+1}$ and $\tilde{\beta}_{k+1} = 0$.

In LMBM both the limited memory BFGS (L-BFGS) and the limited memory SR1 (L-SR1) update formulae [14] are used in calculations of the search direction and the aggregate values. The idea of *limited memory matrix updating* is that instead of storing large $n \times n$ matrices D^k, one stores a certain (usually small), say m_c, number of correction vectors obtained at the previous iterations of the algorithm, and uses these vectors to implicitly define the variable metric matrices. In the case of a null step, we use the L-SR1 update, since this update formula allows us to preserve the boundedness and some other properties of generated matrices which guarantee the global convergence of the method. Otherwise, since these properties are not required after a serious step, the more efficient L-BFGS update is employed (for more details, see [25, 26, 27]).

For smooth functions, a necessary condition for a local minimum is that the gradient has to be zero and by continuity it becomes small when we are close to an optimal point. This is no longer true when we replace the gradient by an arbitrary subgradient. Due to the subgradient aggregation, we have quite a useful approximation to the gradient at our disposal, namely the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$. However, as a stopping criterion, the direct test $\|\tilde{\boldsymbol{\xi}}_k\| < \varepsilon$, for some $\varepsilon > 0$, is too uncertain, if the current piecewise linear approximation of the objective function is too rough. Therefore, we use the term $\tilde{\boldsymbol{\xi}}_k^T D_k \tilde{\boldsymbol{\xi}}_k = -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k$ and the aggregate subgradient locality measure $\tilde{\beta}_k$ to improve the accuracy of $\|\tilde{\boldsymbol{\xi}}_k\|$ (see, e.g., [43]). Hence, the stopping parameter w_k at iteration k is defined by

$$w_k = -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k + 2\tilde{\beta}_k$$

and the algorithm stops if $w_k \leq \varepsilon_S$ for some user specified $\varepsilon_S > 0$. The parameter w_k is also used during the line search procedure to represent the desirable amount of descent.

The pseudo-code of LMBM is the following:

```
PROGRAM LMBM
  INITIALIZE $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$, and $\varepsilon_S > 0$;
  Set $k = 1$ and $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$;
  WHILE the termination condition $w_k \le \varepsilon_S$ is not met
    Find step sizes $t_L^k$ and $t_R^k$;
    Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k$;
    Evaluate $f(\boldsymbol{x}_{k+1})$ and $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{x}_k + t_R^k \boldsymbol{d}_k)$;
    IF $t_L^k > 0$ THEN
      SERIOUS STEP
        Compute the search direction $\boldsymbol{d}_{k+1}$ using $\boldsymbol{\xi}_{k+1}$ and L-BFGS
          update;
      END SERIOUS STEP
    ELSE
      NULL STEP
        Compute the aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k+1}$;
        Compute the search direction $\boldsymbol{d}_{k+1}$ using $\tilde{\boldsymbol{\xi}}_{k+1}$ and L-SR1
          update;
      END NULL STEP
    END IF
    Set $k = k + 1$;
  END WHILE
  RETURN final solution $\boldsymbol{x}_k$;
END LMBM
```

Under the upper semismoothness assumption [11] LMBM can be proved to be globally convergent for LLC objective functions [25, 27].

2.3 Sparse Hessian Approximation

The classical variable metric techniques for nonlinear optimization (see, e.g. [19]) construct a dense $n \times n$ -matrix that approximates the Hessian of the function. These techniques require to store and manipulate this dense matrix, which in large-scale setting becomes unmanageable. In the limited memory variable metric methods (see, e.g. [14, 48]) the storage of this large matrix can be avoided, but still the formed approximation of the Hessian is dense. This is also true for LMBM described in previous subsection.

Nevertheless, in many large-scale problems the real Hessian (if it exists) is sparse. Thus, we would like to compute a variable metric matrix B^{k+1} ($B^{k+1} = (D^{k+1})^{-1}$ in previous subsection) with a *given sparsity pattern*

$$B_{i,j}^{k+1} = 0, \qquad \text{for } (i,j) \in I,$$

where I is a set of pairs of integers that defines the required structure of the matrix.

Sparse variable metric updates have been studied, for instance, in [20, 21, 55]. Here we first recall the approach of Fletcher et al. [21], since it is close to the limited memory approach.

Let us define the bundle of the m_c most recent correction vectors

$$S_k = [\boldsymbol{s}_{k-m_c+1} \dots \boldsymbol{s}_k] \qquad \text{and} \qquad U_k = [\boldsymbol{u}_{k-m_c+1} \dots \boldsymbol{u}_k], \tag{7}$$

where $\boldsymbol{s}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ and $\boldsymbol{u}_k = \nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k-1})$. A new update matrix B^{k+1} is defined by

$$\begin{cases} \text{minimize} & \|B^{k+1}S_k - U_k\|_F^2 \\ \text{subject to} & B^{k+1} = (B^{k+1})^T \\ & B^{k+1}_{i,j} = 0 \text{ for } (i,j) \in I. \end{cases} \tag{8}$$

This minimization problem has a solution with any predefined structure of B^{k+1}. Moreover, similarly to limited memory variable metric methods, it does not require the storage of the previous approximation B^k but only a few correction vectors $\boldsymbol{s}_k$ and $\boldsymbol{u}_k$. However, B^{k+1} is not ensured to be positive definite and it is not necessary easy to check the positive definiteness of the formed matrix.

In [30] Herskovits and Goulart proposed to choose a structure for B^{k+1} such that it is easy to check if it is positive definite. This check is then included as a constraint to problem (8). Particularly, they chose B^{k+1} to be a diagonal matrix. Now a new update matrix B^{k+1} is defined by

$$\begin{cases} \text{minimize} & \|B^{k+1}S_k - U_k\|_F^2 \\ \text{subject to} & B^{k+1}_{i,j} = 0 \text{ for } i \neq j \\ & B^{k+1}_{i,i} \geq \varepsilon \text{ for } i = 1, 2, \dots, n \text{ and } \varepsilon > 0. \end{cases} \tag{9}$$

This minimization problem has a solution

$$B^{k+1}_{i,i} = \begin{cases} \boldsymbol{b}_i / Q_{i,i}, & \text{if } \boldsymbol{b}_i / Q_{i,i} > \varepsilon \\ \varepsilon, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{b} = 2\sum_{i=1}^{m_c} \operatorname{diag}(\boldsymbol{s}_i)\boldsymbol{u}_i$ and $Q_{i,i} = 2\sum_{i=1}^{m_c} [\operatorname{diag}(\boldsymbol{s}_i)]^2$, and $\operatorname{diag}(\boldsymbol{v})$, for $\boldsymbol{v} \in \mathbb{R}^n$, is a diagonal matrix such that $\operatorname{diag}(\boldsymbol{v})_{i,i} = \boldsymbol{v}_i$.

3 Diagonal Bundle Method

In this section we introduce the new diagonal bundle method (D-Bundle) for sparse nonsmooth minimization. The basic idea of D-Bundle is to simplify

LMBM and use sparse, positive definite, updates. Further, we will prove the global convergence of the method for LLC functions. As with LMBM we assume that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC and we can compute $f(\boldsymbol{x})$ and $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ at every $\boldsymbol{x} \in \mathbb{R}^n$.

3.1 D-Bundle

Similarly to LMBM, D-Bundle uses m_c most recent correction vectors to compute updates for matrices. These vectors are defined by

$$S_k = [\boldsymbol{s}_{k-m_c+1} \dots \boldsymbol{s}_k] \qquad \text{and} \qquad U_k = [\boldsymbol{u}_{k-m_c+1} \dots \boldsymbol{u}_k],$$

where $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ with $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ and $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$. Note that, due to fact that the gradient does not need to exist for nonsmooth objective, these correction vectors are computed using subgradients (cf. eq. (7)). Moreover, due to usage of null steps we may have $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and thus, we use here the auxiliary point $\boldsymbol{y}_{k+1}$ instead of $\boldsymbol{x}_{k+1}$.

The obvious difference between D-Bundle and LMBM is that we now use the diagonal update formula (9) to compute the *diagonal variable metric update*. Therefore, the search direction is computed by the formula

$$\boldsymbol{d}_k = -D^k \tilde{\boldsymbol{\xi}}_k, \tag{10}$$

where $\tilde{\boldsymbol{\xi}}_k$ is (an aggregate) subgradient and D^k represents the diagonal update matrix such that $D^k = (B^k)^{-1}$ in (9). Naturally, the weakness of using this approach is that the real sparsity pattern of the problem is not used. However, using diagonal update matrix requires minimum amount of storage space and computations, and as stated in the previous section, the positive definiteness of generated matrices can be easily guaranteed.

To ensure the global convergence of D-Bundle, we have to assume that all the matrices D^k are bounded. Due to our diagonal update formula this assumption is trivially satisfied. In addition, the condition

$$\tilde{\boldsymbol{\xi}}_k^T D^k \tilde{\boldsymbol{\xi}}_k \le \tilde{\boldsymbol{\xi}}_k^T D^{k-1} \tilde{\boldsymbol{\xi}}_k \tag{11}$$

has to be satisfied each time there occurs more than one consecutive null step. In D-Bundle this is guaranteed simply by *skipping the updates*. That is, after a null step we set $D^{k+1} = D^k$, but the new aggregate values are computed.

D-Bundle uses the *aggregation procedure* similar to LMBM to guarantee the convergence of the method and to avoid unbounded storage — the convex combination of at most three subgradients is used to form a new aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k+1}$ and a new aggregate subgradient locality measure $\tilde{\beta}_{k+1}$ (cf. (5) and (6)). Of course, the diagonal update matrix D^k is used in equation (5) instead of limited memory update.

Furthermore, the following properties are kept as in LMBM:

- *Serious and null steps:* in D-Bundle the line search procedure (cf. (1) – (3)) is used to determine a new iteration and auxiliary points $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_{k+1}$. That is, the step sizes $t_R^k \in (0, t_{max}]$ and $t_L^k \in [0, t_R^k]$ with $t_{max} > 1$ are computed such that either condition (2) for serious steps or condition (3) for null steps is satisfied.
- *Stopping criterion:* as a stopping parameter D-Bundle uses the value

$$w_k = -\tilde{\boldsymbol{\xi}}_k^T \boldsymbol{d}_k + 2\tilde{\beta}_k \qquad (12)$$

and stops if $w_k \leq \varepsilon_S$ for some user specified $\varepsilon_S > 0$. Similarly to LMBM, the parameter w_k is used also during the line search procedure to represent the desirable amount of descent (cf. (2) and (3)).

The pseudo-code of D-Bundle is the following (note that obviously more details are given here than before):

```
PROGRAM D-BUNDLE
  INITIALIZE $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$, $m_c \geq 1$ and $\varepsilon_S > 0$;
  Set $k = 1$, $m = 1$, $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$, $\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$, and $\tilde{\beta}_1 = 0$;
  WHILE the termination condition $w_k \leq \varepsilon_S$ is not met
    Find step sizes $t_L^k$ and $t_R^k$, and the subgradient
      locality measure $\beta_{k+1}$;
    Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k$ and $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k$;
    Evaluate $f(\boldsymbol{x}_{k+1})$ and $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$;
    IF $t_L^k > 0$ THEN
      SERIOUS STEP
        Store the new correction vectors $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$
          and $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$;
        Compute the new diagonal matrix $D^{k+1}$ using $m_c$ most
          recent correction vectors;
        Compute the search direction $\boldsymbol{d}_{k+1} = -D^{k+1}\boldsymbol{\xi}_{k+1}$;
        Set $m = k + 1$ and $\tilde{\beta}_{k+1} = 0$;
      END SERIOUS STEP
    ELSE
      NULL STEP
        Compute the aggregate values
          $\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k$    and
          $\tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k$;
        Set $D^{k+1} = D^k$;
        Compute the search direction $\boldsymbol{d}_{k+1} = -D^{k+1}\tilde{\boldsymbol{\xi}}_{k+1}$;
      END NULL STEP
    END IF
    Set $k = k + 1$;
  END WHILE
  RETURN final solution $\boldsymbol{x}_k$;
END D-BUNDLE
```

3.2 Convergence Analysis

We now prove the global convergence of the D-BUNDLE-algorithm. In addition to assuming that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC, the level set $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ is supposed to be bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Furthermore, we assume that each execution of the line search procedure is finite. The line search procedure has been proved to be finite under the assumption of upper semismoothness (see [57]).

The convergence analysis of D-BUNDLE is very similar to that of LMBM. In fact, all the results in [27] are valid also for D-BUNDLE except that, for D-

Bundle, we do not have property $\boldsymbol{u}_k^T(D^k\boldsymbol{u}_k - \boldsymbol{s}_k) > 0$ given in Lemma 1 of [27]. With LMBM this property is used to guarantee that condition (11) is valid in the case of consecutive null steps. However, with D-Bundle condition (11) is valid due to skipping of updates and, thus, the above mentioned property is not required.

We start the theoretical analysis of D-Bundle by showing that the generated matrices are bounded. We say that a matrix is bounded if its eigenvalues lie in a compact interval not containing zero. After that we prove that, if D-Bundle stops at iteration k, then the point $\boldsymbol{x}_k$ is a stationary point of f. Further, we prove that in the case of an infinite sequence $\{\boldsymbol{x}_k\}$, every accumulation point $\bar{\boldsymbol{x}}$ of the sequence $\{\boldsymbol{x}_k\}$ generated by the algorithm is a stationary point of f.

Remark 3.1. The sequence $\{\boldsymbol{x}_k\}$ generated by D-Bundle is bounded by assumption and the monotonicity of the sequence $\{f_k\}$ which, in turn, is obtained due to condition (2) 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 and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| \leq t_{max}C$ for null steps by equation (1) 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 (see [27] for more details). By the local boundedness and the upper semi-continuity of the subdifferential, we obtain the boundedness of subgradients $\boldsymbol{\xi}_k$ as well as their convex combinations (see [16]).

Lemma 3.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a LLC function and suppose that the level set $\{\,\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1)\,\}$ is bounded. Then the matrix D^k is bounded for all $k \geq 1$. In addition, in the case of a null step, we have $w_{k+1} \leq w_k$ for all $k \geq m$.*

Proof. The boundedness of D^k follows directly from Remark 3.1 and the fact that $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ with $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ and $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$.

In the case of a null steps, we have

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

for all $k \geq m$ by (5), (6), (10) and (12), and by the fact that we always set $D^{k+1} = D^k$ in the case of a null step. □

Theorem 3.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a LLC function and suppose that the level set $\{\,\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1)\,\}$ is bounded. If* D-Bundle *stops with a finite number of* *iterations, then the solution $\boldsymbol{x}_k$ is a stationary point of f. On the other hand, any accumulation point of an infinite sequence of solutions generated by* D-Bundle *is a stationary point of f.*

Proof. In order to prove this proposition we assume $\varepsilon_S = 0$. D-Bundle algorithm is essentially similar to LMBM with the diagonal matrix updating instead of limited memory matrix updating. According to convergence analysis of LMBM (see [27]), if the algorithm stops with a finite number of iterations, then $\mathbf{0} \in \partial f(\boldsymbol{x}_k)$. This result remains same for D-Bundle. If the algorithm generates an infinite sequence of solutions, then we can replace Lemma 7 in [27] and the first part of the proof of Lemma 8 in [27] by Lemma 3.1 and all the remaining results of [27] are valid also for D-Bundle.

Therefore, similarly to LMBM, D-Bundle either terminates at a stationary point of the objective function f or generates an infinite sequence $(\boldsymbol{x}_k)$ for which accumulation points are stationary for f. Moreover, if we choose $\varepsilon > 0$, the D-Bundle method terminates in a finite number of steps. □

4 Numerical Experiments

In this section we compare D-Bundle with some existing methods for NSO. The test set used in our experiments consists of extensions of classical academic nonsmooth minimization problems [26]. We have tested these problems up to million variables.

4.1 Solvers

The tested optimization codes with references to more detailed descriptions of the methods and their implementations are presented in Table 1.

Table 1: Tested pieces of software

Software	Author(s)	Method	Reference
`PBNCGC`	Mäkelä	Proximal bundle	[42, 43]
`QSM`	Bagirov & Ganjehlou	Quasi-Secant	[5, 6]
`LMBM`	Karmitsa	Limited memory bundle	[26, 27]
`D-Bundle`	Karmitsa	Diagonal bundle	
`I-Bundle`	Karmitsa	Diagonal bundle	

A brief description of each software and the references from where the code can be downloaded are in order.

PBNCGC is an implementation of the most frequently used bundle method in NSO; that is, the proximal bundle method. The code includes constraint handling (bound constraints, linear constraints, and nonlinear/nonsmooth constraints) and a possibility to optimize multiobjective problems. The quadratic direction finding problem characteristic for bundle methods is solved by the PLQDF1 subroutine implementing the dual projected gradient method proposed in [39].

PBNCGC can be used (free for academic purposes) via WWW-NIMBUS -system (http://nimbus.mit.jyu.fi/) [44]. Furthermore, the Fortran 77 source code is available for downloading from http://napsu.karmitsa.fi/proxbundle/.

QSM is a quasi-secant solver for nonsmooth possibly nonconvex minimization. Although originally developed for small- and medium-scale problems, QSM has shown to be very efficient also in large-scale setting [7, 33].

The user can employ either analytically calculated or approximated subgradients in his experiments (this can be done automatically by selecting one parameter). We have used analytically calculated subgradients here.

The Fortran 77 source code is available for downloading from http://napsu.karmitsa.fi/qsm/.

LMBM is an implementation of a limited memory bundle method specifically developed for large-scale NSO. In our experiments, we used the adaptive version of the code with the initial number of stored correction pairs used to form the variable metric update equal to 7 and the maximum number of stored correction pairs equal to 15.

The Fortran 77 source code and the mex-driver (for MatLab users) are available for downloading from http://napsu.karmitsa.fi/lmbm/.

D-Bundle is an implementation of the diagonal bundle method introduced in this paper. The Fortran 95 source code of D-Bundle is available for downloading from http://napsu.karmitsa.fi/dbundle/.

I-Bundle is the same code as D-Bundle but instead of updating diagonal matrix D^k and using the correction pairs, we use the identity matrix I in our computations. The source code of D-Bundle includes I-Bundle.

All of the algorithms except for D- and I-Bundles were implemented in Fortran77 using double-precision arithmetic. The experiments were performed on an Intel® Core™ 2 CPU 1.80GHz. To compile the codes, we used gfortran, the GNU Fortran compiler.

4.2 Test problems and parameters

As already said the test set used in our experiments consists of extensions of classical academic nonsmooth minimization problems from the literature. That is problems 1 – 10 first introduced in [26]. Problems 1 – 5 are convex while problems 6 – 10 are nonconvex. These problems can be formulated with any number of variables. We have used here 1000, 10 000, 100 000 and 1 000 000 variables.

To obtain comparable results the stopping parameters of the codes were tuned by the procedure similar to [7]. In addition to the usual stopping criteria of the solvers, we terminated the experiments if the elapsed CPU time exceeded half an hour for problems with 1000 variables, an hour with 10 000 variables, and two hours with 100 000 and million variables.

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

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

where f_{best} is a solution obtained with the solver and f_{opt} is the best known (or optimal) solution. We have *accepted the results* with respect to the tolerance $\varepsilon = 10^{-3}$. In addition, we say that the result is *inaccurate*, if a solver finds the solution with respect to a tolerance $\varepsilon = 10^{-2}$. Otherwise, we say that a solver *fails*.

For LMBM, PBNCGC and QSM the maximum size of the bundle was set to 100. With D-Bundle and I-Bundle the natural choice for the bundle size is *two*. For all other parameters we have used the default settings of the codes. For D- and I-Bundles these are

$$\varepsilon = 1, \qquad \varepsilon_L = 10^{-4}, \qquad \varepsilon_R = 0.25,$$

and

$$t_{max} = \begin{cases} 1000.0 & \text{for convex } f, \\ 1.5 & \text{for nonconvex } f, \end{cases} \qquad \gamma = \begin{cases} 0.1 & \text{for convex } f, \\ 1.0 & \text{for nonconvex } f. \end{cases}$$

In addition, for D-Bundle the number of stored correction pairs was set to *three* and, naturally, with I-Bundle it was *zero*.

4.3 Results

The results are summarized in Tables 2 – 5. We have compared the efficiency of the solvers both in terms of the computational time (*cpu*) and the number of function and subgradient evaluations (nfg, evaluations for short). We have used bold-face text to emphasize the best results.

Table 2: Summary of the results with 1000 variables.

P	`PBNCGC`	`QSM`	`LMBM`	`D-bundle`	`I-bundle`
	nfg/cpu	nfg/cpu	nfg/cpu	nfg/cpu	nfg/cpu
1	19 738/789.59	18 263/7.19	fail	**6 136**/0.60	**6 136**/**0.44**
2	**46/0.57**	4 242/87.67	fail	inacc/6.47	fail
3	24 424/1 800.00	2 326/4.60	6 540/0.32	**242/0.01**	**242**/0.02
4	16 388/1 800.13	2 036/0.88	**558/0.37**	6 843/1.12	7 910/1.55
5	**56**/0.07	667/0.10	228/**0.04**	3 643/0.66	3 542/0.65
6	272/0.05	**254/0.03**	1 062/0.94	1 126/0.20	1 126/0.10
7	**76/0.22**	inacc/0.93	352/0.37	6 119/3.24	fail
8	28 318/1 800.09	2 836/6.83	**1 230**/0.67	7 974/0.44	3 786/**0.13**
9	**98**/0.10	inacc/0.02	200/0.06	569/0.05	569/**0.03**
10	**398/6.73**	inacc/11.74	inacc/0.31	fail	fail

Table 3: Summary of the results with 10 000 variables.

P	`PBNCGC`	`QSM`	`LMBM`	`D-bundle`	`I-bundle`
	nfg/cpu	nfg/cpu	nfg/cpu	nfg/cpu	nfg/cpu
1	fail	**160 910**/1 000.62	fail	179 502/214.53	179 502/**176.34**
2	**54/56.06**	fail	fail	fail	fail
3	13 026/3 600.21	1 868/5.20	**796**/7.45	2 161/**0.99**	2 161/1.14
4	5 960/3 600.83	2086/**7.32**	**882**/8.81	fail	fail
5	**54/0.53**	903/1.41	784/3.00	3 370/5.47	3200/4.76
6	**74/0.26**	inacc/0.35	10 106/143.32	10 102/9.09	10 102/9.09
7	**244/23.36**	inacc/15.39	inacc/9.68	fail	fail
8	8 826/3 600.48	2 383/8.53	**1 194**/6.11	5 311/**1.98**	5 311/**1.98**
9	**284**/2.20	fail	396/3.20	575/**0.34**	575/**0.34**
10	**2472/811.77**	inacc/164.68	fail	inacc/2.45	inacc/2.45

The results for problems with 1000 and 10 000 variables reveal similar trends (see Tables 2 and 3)): In both cases PBNCGC was clearly the most robust solver. In addition, PBNCGC usually used either the least or the most evaluations, making it very difficult to say if it is an efficient method or not. The robustnesses of the new solvers D-Bundle and I-Bundle were similar to LMBM and QSM.

The results of D- and I-Bundles were very similar. However, we may say that I-Bundle was usually a little bit more efficient but less robust than D-Bundle.

Both D-Bundle and I-Bundle usually used more evaluations than LMBM. However, in terms of cpu-time they were comparable or — especially with larger problem — even better than LMBM. Furthermore, both D- and I-Bundles succeed in solving P1 which has shown to be very difficult to LMBM due its sparse structure.

Table 4: Summary of the results with 100 000 variables.

P	QSM	LMBM	D-bundle	I-bundle
	nfg/cpu	nfg/cpu	nfg/cpu	nfg/cpu
1	fail	fail	fail	fail
2	fail	fail	fail	fail
3	1 687/46.39	1 718/184.16	**145**/0.99	**145**/**0.73**
4	2 137/**77.20**	**1 258**/147.37	fail	fail
5	1 252/111.28	802/63.27	827/15.44	**542**/**7.05**
6	fail	fail	fail	fail
7	inacc/254.81	fail	**3 152**/**552.43**	41 539/785.48
8	**1 237**/**26.09**	2 300/245.10	3 810/38.67	6 598/36.63
9	inacc/83.64	**994**/99.41	1 159/15.29	1159/**9.55**
10	inacc/93.09	inacc/157.20	inacc/23.22	fail

A problem with 100 000 variables can be considered as an extremely large nonsmooth problem. With the problems of this size, the solver PBNCGC could not be compiled due to the arithmetic overflow.

Moreover, the other solvers succeed in solving at most five problems out of ten (see Table 4). Thus, no very far-reaching conclusions can be made. However, as before, the robustnesses of D-Bundle and I-Bundle were similar to LMBM and QSM, and D-Bundle was a little bit more robust than I-Bundle. In addition, D-Bundle and I-Bundle usually used clearly less cpu-time than LMBM and QSM.

With million variable QSM could not be compiled either. Furthermore, the solver LMBM with the bundle size equal to 100 could be compiled but not run: the procedure was killed by the host for using too many resources. Thus, for million variables we changed the size of the bundle to *two* also for LMBM.

Now, D-Bundle was the most robust solver. It succeed in solving six problems out of ten with the desired accuracy while both I-Bundle and LMBM succeed in

Table 5: Summary of the results with million variables.

P	LMBM nfg/cpu	D-bundle nfg/cpu	I-bundle nfg/cpu
1	fail	fail	fail
2	fail	fail	fail
3	**1 698**/168.18	2 431/155.81	2 431/**135.41**
4	14 074/2 212.79	7 120/1 397.11	**7 060/1 359.31**
5	1 692/414.81	**371/70.72**	416/78.08
6	fail	fail	fail
7	fail	**6 258/2 896.71**	inacc/7 200.48
8	**4 970**/924.28	6 872/742.20	6 583/**511.81**
9	3 702/869.06	**3 319**/540.67	**3319/359.20**
10	fail	inacc/7201.68	inacc/7202.48

solving five problems. In addition, with `D-` and `I-Bundle`s some of these failures were inaccurate result: with the relaxed tolerance $\varepsilon = 10^{-2}$ they succeed in solving seven problems while with `LMBM` the number of failures is still five even if the relaxed tolerance was used. `I-Bundle` usually used less cpu-time than `D-Bundle`, while the numbers of evaluations were about the same.

5 Conclusions

In this paper, we have described a new diagonal bundle method (D-Bundle) for unconstrained nonsmooth optimization. We have proved the global convergence of the method for locally Lipschitz continuous semismooth objective functions, which are not necessarily differentiable or convex.

The numerical experiments reported confirm that D-Bundle is efficient for both convex (problems 1 – 5) and nonconvex (problems 6 – 10) nonsmooth optimization problems. With large problems ($n \leq 1\,000$) D-Bundle was comparable with the existing solvers. With larger numbers of variables ($n = 10\,000$ or $100\,000$) D-Bundle usually used clearly less cpu-time than the other solvers tested. In addition, with extremely large numbers of variables ($n \geq 1000\,000$) D-Bundle was the best solver tested.

We can conclude that D-Bundle is a good alternative to existing nonsmooth optimization algorithms and for extremely large-scale problems it might well be the best choice available.

Acknowledgments

The work was financially supported by the University of Turku (Finland) and Magnus Ehrnrooth foundation.

References

[1] Apkarian, P., Noll, D., and Prot, O. A trust region spectral bundle method for non-convex eigenvalue optimization. *SIAM Journal on Optimization 19*, 1 (2008), 281–306.

[2] Astorino, A., and Fuduli, A. Nonsmooth optimization techniques for semi-supervised classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 12 (2007), 2135–2142.

[3] Astorino, A., Fuduli, A., and Gorgone, E. Nonsmoothness in classification problems. *Optimization Methods and Software 23*, 5 (2008), 675–688.

[4] Äyrämö, S. *Knowledge Mining Using Robust Clustering*. PhD thesis, University of Jyväskylä, Department of Mathematical Information Technology, 2006.

[5] Bagirov, A. M., and Ganjehlou, A. N. A secant method for nonsmooth optimization. Submitted, 2009.

[6] Bagirov, A. M., and Ganjehlou, A. N. A quasisecant method for minimizing nonsmooth functions. *Optimization Methods and Software 25*, 1 (2010), 3–18.

[7] Bagirov, A. M., Karmitsa, N., and M. M. M. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014. to appear.

[8] Beck, A., and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters 31*, 3 (2003), 167–175.

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

[10] Bergeron, C., Moore, G., Zaretzki, J., Breneman, C., and Bennett, K. Fast bundle algorithm for multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012).

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

[12] Bradley, P. S., Fayyad, U. M., and Mangasarian, O. L. Mathematical programming for data mining: Formulations and challenges. *INFORMS Journal on Computing 11* (1999), 217–238.

[13] Burke, J. V., Lewis, A. S., and Overton, M. L. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization 15* (2005), 751–779.

[14] 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.

[15] Carrizosa, E., and Romero Morales, D. Supervised classification and mathematical optimization. *Computers and Operations Research 40*, 1 (2013), 150–165.

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

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

[18] Demyanov, V. F., Bagirov, A. M., and Rubinov, A. M. A method of truncated codifferential with application to some problems of cluster analysis. *Journal of Global Optimization 23*, 1 (2002), 63–80.

[19] Fletcher, R. *Practical Methods of Optimization*, 2nd edition ed. John Wiley & Sons, Chichester, 1987.

[20] Fletcher, R. An optimal positive definite update for sparse Hessian matrices. *SIAM Journal on Optimization 5*, 1 (1995), 192–218.

[21] Fletcher, R., Grothey, A., and Leyffer, S. Computing sparse Hessian and Jacobian approximations with optimal hereditary properties. University of Dundee Numerical Analysis Report NA/164, 1995.

[22] Fuduli, A., Gaudioso, M., and Giallombardo, G. Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. *SIAM Journal on Optimization 14*, 3 (2004), 743–756.

[23] Fuduli, A., Gaudioso, M., and Nurminski, E. A. A splitting bundle approach for non-smooth non-convex minimization. Optimization, in press, DOI: 10.1080/02331934.2013.840625, 2013.

[24] Gaudioso, M., and Monaco, M. F. Variants to the cutting plane approach for convex nondifferentiable optimization. *Optimization 25* (1992), 65–75.

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

[26] 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.

[27] 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.

[28] Hare, W., and Sagastizábal, C. A redistributed proximal bundle method for nonconvex optimization. *SIAM Journal on Optimization 20*, 5 (2010), 2442–2473.

[29] Haslinger, J., and Neittaanmäki, P. *Finite Element Approximation for Optimal Shape, Material and Topology Design*, 2nd edition ed. John Wiley & Sons, Chichester, 1996.

[30] Herskovits, J., and Goulart, E. Sparse quasi-Newton matrices for large scale nonlinear optimization. In *Proceedings of the 6th Word Congress on Structurla and Multidisciplinary Optimization* (2005).

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

[32] Kärkkäinen, T., and Heikkola, E. Robust formulations for training multilayer perceptrons. *Neural Computation 16* (2004), 837–862.

[33] Karmitsa, N., Bagirov, A., and Mäkelä, M. M. Comparing different nonsmooth optimization methods and software. *Optimization Methods and Software 27*, 1 (2012), 131–153.

[34] Karmitsa, N., Mäkelä, M. M., and Ali, M. M. Limited memory interior point bundle method for large inequality constrained nonsmooth minimization. *Applied Mathematics and Computation 198*, 1 (2008), 382–400.

[35] Karmitsa, N., Tanaka Filho, M., and Herskovits, J. Globally convergent cutting plane method for nonconvex nonsmooth minimization. *Journal of Optimization Theory and Applications* (2011).

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

[37] Lemaréchal, C. Nondifferentiable optimization. In *Optimization*, G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, Eds. Elsevier North-Holland, Inc., New York, 1989, pp. 529–572.

[38] 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. 245–281.

[39] Lukšan, L. Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minmax approximation. *Kybernetika 20* (1984), 445–457.

[40] 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.

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

[42] 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.

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

[44] Miettinen, K., and Mäkelä, M. M. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research 170*, 3 (2006), 909–922.

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

[46] 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.

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

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

[49] Noll, D., Prot, O., and Rondepierre, A. A proximity control algorithm to minimize nonsmooth and nonconvex functions. *Pacific Journal of Optimization 4*, 3 (2008), 571–604.

[50] 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.

[51] Polak, E., and O. R. J. Algorithms for finite and semi-finite min-max-min problems using adaptive smoothing techniques. *Journal of Optimization Theory and Applications 119* (2003), 421–457.

[52] Sagastizábal, C., and Solodov, M. An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. *SIAM Journal on Optimization 16*, 1 (2005), 146–169.

[53] 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.

[54] Shor, N. Z. *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, Berlin, 1985.

[55] Toint, P. L. On sparse and symmetric matrix updating subject to a linear equation. *Mathematics of Computation 31*, 140 (1977), 954–961.

[56] Uryasev, S. P. Algorithms for nondifferentiable optimization problems. *Journal of Optimization Theory and Applications 71* (1991), 359–388.

[57] 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.