To appear in *Optimization Methods & Software*
Vol. 00, No. 00, Month 20XX, 1–20

Diagonal Bundle Method with Convex and Concave Updates for Large-Scale Nonconvex and Nonsmooth Optimization

N. Karmitsa[a*] and M. Gaudioso[b] and K. Joki[a]

[a]*Department of Mathematics and Statistics, University of Turku, FI-20014 Turku, Finland;*
[b]*Dipartimento di Elettronica e Sistemistica, Universitá della Calabria, 87036, Rende (CS), Italy*

(*Submitted: April 2017*)

Nonsmooth optimization is traditionally based on convex analysis and most solution methods rely strongly on the convexity of the problem. In this paper, we propose an efficient diagonal bundle method for nonconvex large-scale nonsmooth optimization. The novelty of the new method is in different usage of metrics depending on the convex or concave behaviour of the objective at the current iteration point. The usage of different metrics gives us a possibility to better deal with the nonconvexity of the problem than the sole — the most commonly used and quite arbitrary — downward shifting of the piecewise linear model does. The convergence of the proposed method is proved for semismooth functions that are not necessarily differentiable nor convex. The numerical experiments have been made using problems with up to one million variables. The results to be presented confirm the usability of the new method.

Keywords: Nondifferentiable optimization; nonconvex problems; bundle methods, diagonal variable metric updates

AMS Subject Classification: 65K05; 90C06; 90C26; 90C53

1. Introduction

Nonsmooth optimization (NSO) refers to the general problem of minimizing (or maximizing) functions that have discontinuous gradients (see e.g., [5]). NSO problems are encountered in many application areas: for instance, in economics [41], mechanics [39], engineering [38], control theory [14], optimal shape design [24], machine learning [27], and data mining [4, 9] including cluster analysis [6, 15, 30, 31] and classification [2, 3, 7, 12]. Most of these problems are large-scale and nonconvex.

In this paper, we are consider solving the problem 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 semismooth and the number of variables n is supposed to be large. Note that no differentiability or convexity assumptions for the problem (P) are made.

NSO problems are in general difficult to solve even when the size of the problem is

*Corresponding author. Email: napsu@karmitsa.fi

small. In addition, besides problematics of nonsmoothness and the size of the problem, nonconvexity adds another challenge; NSO is traditionally based on convex analysis and most solution methods rely strongly on the convex model of the problem. Fortunately, several nonconvex algorithms have been introduced only recently, for instance, in [1, 10, 17–20, 22, 23, 32, 40]. Nevertheless, most of these algorithms are developed only for small-scale problems.

The convex model of the objective function is usually reasonably good for nonconvex problems as well, except in some areas where there exists so-called concave behaviour in the objective. In these cases the linearization error, used as a measure of the goodness of the current piecewise linear model, has negative values and the model is no longer an underestimate of the objective. The common way to deal with this difficulty is to do some downward shifting (e.g. to use the so-called subgradient locality measures instead of linearization errors), but the amount of this shifting may be more or less arbitrary. In [17, 20] the concave behaviour in the objective is somewhat better minded. The contribution to the model of points characterized by positive or negative values of the linearization error is kept separate, which results in the need to maintain two distinct "bundles" of information.

In this paper, we introduce a new *splitting metrics diagonal bundle algorithm* (SMDB) for solving general, nonconvex, large-scale NSO problems. The SMDB combines the ideas of the *diagonal bundle method* (D-Bundle, [28]) to different usage of metrics depending on the convex or concave behaviour of the objective at the current iteration point. The D-Bundle, in turn, is developed for sparse large-scale nonsmooth, possible nonconvex, optimization. It is a successor of the *limited memory bundle method* (LMBM, [21, 22]) and the *variable metric bundle method* (VMBM, [35, 42]) and better capable of handling large dimensionality and sparsity of the objective. In the D-Bundle the nonconvexity is taken into account by means of subgradient locality measures. The idea of splitting the data with the SMDB comes from [17, 20]. However, instead of splitting the bundle information (i.e. the subgradient information) we now compute different variable metric approximations depending on the point.

The SMDB shares the good properties of the D-Bundle. That is, the time-consuming quadratic direction finding problem appearing in standard bundle methods (see eq. [26, 33, 36]) need not be solved, nor does the number of stored subgradients need to grow with the dimension of the problem. Furthermore, the method uses only a few vectors to represent the diagonal variable metric approximation of the Hessian matrix. Thus, it avoids storing and manipulating large matrices, as in the VMBM, and using dense approximations to the Hessian, as in the LMBM. The usage of different metrics in the SMDB gives us a possibility to better deal with the nonconvexity of the problem than the sole usual subgradient locality measure used in the D-Bundle does.

This paper is organized as follows. In Section 2, we introduce our notation and recall some basic definitions and results from nonsmooth analysis. In Section 3, we discuss the basic ideas of the SMDB and, in Section 4, we prove its convergence. The results of the numerical experiments are presented and discussed in Section 5 and Section 6 concludes the paper.

2. Notations and Background

In this section, we give our notation and recall some basic definitions and results from nonsmooth analysis.

We denote by $\|\cdot\|$ the Euclidean norm in $\mathbb{R}^n$ and by $\boldsymbol{a}^\top \boldsymbol{b}$ the inner product of vectors

$\boldsymbol{a}$ and $\boldsymbol{b}$ (bolded symbols are used for vectors). In addition, we denote by $\mathrm{diag}(\boldsymbol{a})$, for $\boldsymbol{a} \in \mathbb{R}^n$, the diagonal matrix such that $\mathrm{diag}(\boldsymbol{a})_{i,i} = a_i$. The Frobenius norm of a matrix $A \in \mathbb{R}^{n \times n}$ is denoted by $\|A\|_F$. That is, we define

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

The *subdifferential* $\partial f(\boldsymbol{x})$ [13] of a locally Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}$ at any point $\boldsymbol{x} \in \mathbb{R}^n$ is given by

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

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 $\boldsymbol{0} \in \partial f(\boldsymbol{x}^*)$. 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 $k \to \infty$.

3. Splitting Metrics Diagonal Bundle Method

In this section, we introduce a new splitting metrics diagonal bundle algorithm SMDB for solving general, nonconvex, large-scale NSO problems. We assume that at every point $\boldsymbol{x} \in \mathbb{R}^n$ we can evaluate the objective function $f(\boldsymbol{x})$ and obtain one arbitrary subgradient $\boldsymbol{\xi}$ from the subdifferential $\partial f(\boldsymbol{x})$.

We start this section with a simplified flowchart of the new method (in Figure 1) to point out the basic ideas. The SMDB is characterized by the usage of null steps together with the aggregation of subgradients. Moreover, the search direction is calculated by using diagonal variable metric updates. Two alternative update rules can be selected, according to the "local convex" or "local concave" behaviour of the objective function identified by the linearization error. Using null steps gives sufficient information about the nonsmooth objective function in case the current seach direction is not good enough. On the other hand, a simple aggregation of subgradients guarantees the convergence of the aggregate subgradients to zero and makes it possible to evaluate a termination criterion.

Now, we first describe in more detail the different components of the method and then introduce the entire algorithm.

Linearization error. As already stated, the SMDB uses the sign of the linearization error to detect the "convex" or "concave" behaviour of the objective. The linearization error α_{k+1} associated with point $\boldsymbol{y}_{k+1}$ is given by

$$\alpha_{k+1} = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + \boldsymbol{\xi}_{k+1}^\top \boldsymbol{d}_k,$$

where $\boldsymbol{x}_k$ is the current iteration point, $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$ is a new auxiliary point, $\boldsymbol{d}_k$ is the current search direction and $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$. In particular, we say that point $\boldsymbol{y}_{k+1}$ exhibits a "convex"or "concave" behaviour w.r.t. $\boldsymbol{x}_k$, according to the positive or negative sign of α_{k+1}, respectively.

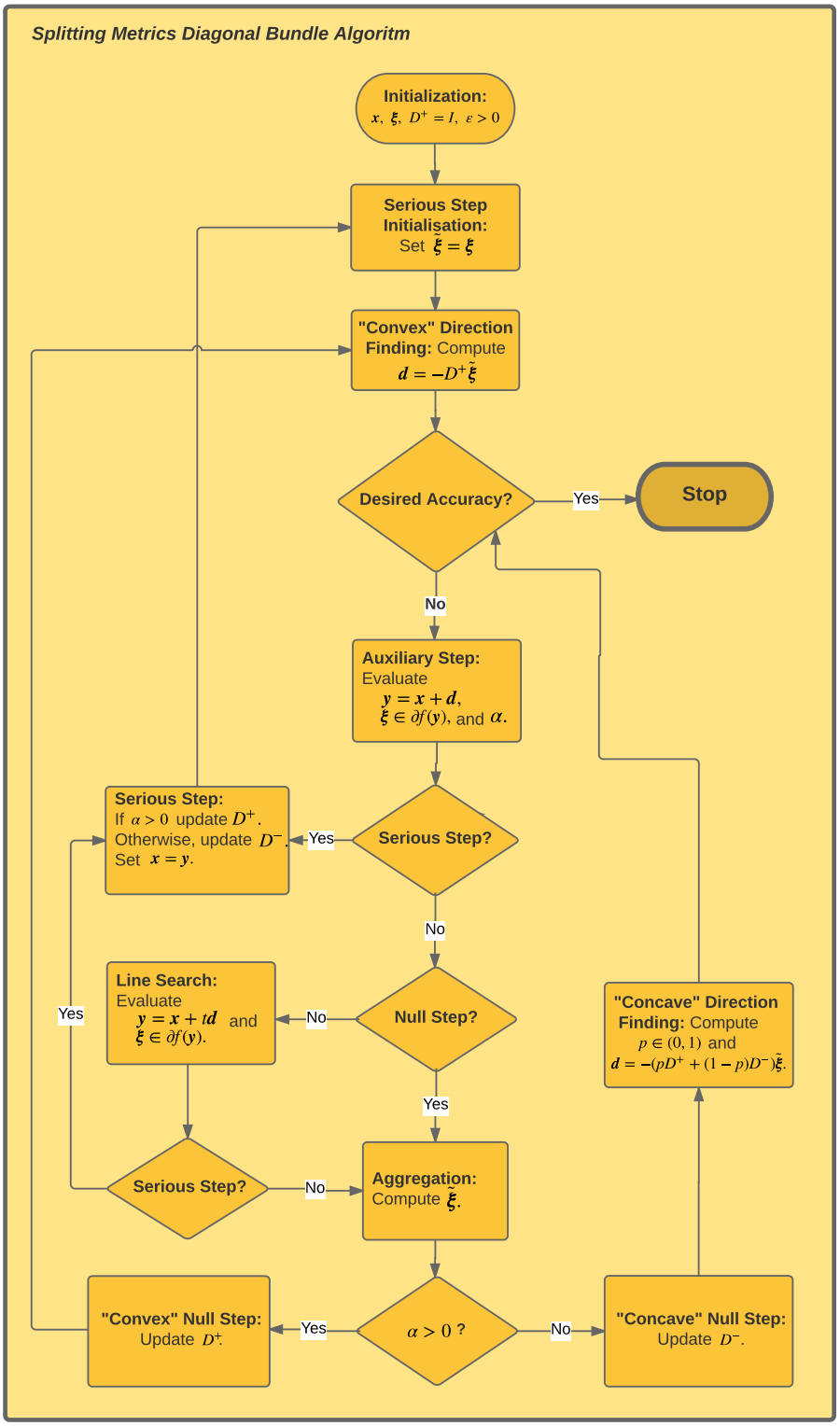


Figure 1. Flowchart of the SMDB. Here, $\tilde{\xi}$ is an aggregate subgradient, D^+ and D^- are the convex and the concave update matrices, respectively, α is the linearization error, and ε is the stopping tolerance.

Matrix Updating and Splitting of Data. We use the diagonal update formula introduced in [25] for updating the matrices in the SMDB, since for this formula it is easy to check and guarantee the positive (negative) definiteness of generated matrices. Moreover, using a diagonal update matrix requires minimum amount of storage space and computations.

A maximum number, say $2m_c$, of correction vectors is used by the SMDB to compute updates for matrices. These correction vectors are quite similar to those in the classical limited memory variable metric methods for smooth optimization (see, e.g. [11]). That is, the correction vectors are given by $\boldsymbol{s}_k = \boldsymbol{y}_{k+1} - \boldsymbol{x}_k$ 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 the fact that the gradient does not need to exist for nonsmooth objective, the correction vectors are computed using subgradients. In addition, due to usage of null steps we may have $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$. Thus, we use here the auxiliary point $\boldsymbol{y}_{k+1}$ instead of $\boldsymbol{x}_{k+1}$. Furthermore, instead of just one couple of correction matrices $S_k = [\boldsymbol{s}_{k-m_c+1} \dots \boldsymbol{s}_k]$ and $U_k = [\boldsymbol{u}_{k-m_c+1} \dots \boldsymbol{u}_k]$ used in [11] and in the D-Bundle [28], we now use different corrections matrices depending on the sign of the linearization error. That is, we append $\boldsymbol{s}_k$ and $\boldsymbol{u}_k$ to S_k^+ and U_k^+, if $\alpha_k \geq 0$ and to S_k^- and U_k^-, otherwise. This means that in each matrix S_k^+ (U_k^+) and S_k^- (U_k^-) we have (at most) m_c correction vectors $\boldsymbol{s}_{\hat{i}}$ ($\boldsymbol{u}_{\hat{i}}$) with indices $\hat{i} \in \{1, 2, \dots, k\}$, and no $\hat{i}$ can be in both S_k^+ (U_k^+) and S_k^- (U_k^-). For simplicity, we will from now on denote these indices by $\hat{i} \in \{\hat{1}, \dots, \hat{m}_c\}$. Note that $\hat{m}_c$ may be smaller than m_c and $\hat{m}_c$ may be different for S_k^+ (U_k^+) and S_k^- (U_k^-).

The approximation of the Hessian B_{k+1}^+ (B_{k+1}^-) is chosen to be a diagonal matrix and the check of positive (negative) definiteness is included as a constraint into the problem. Thus, the update matrix B_{k+1}^+ is defined by

$$\begin{cases} \text{minimize} & \|B_{k+1}^+ S_k^+ - U_k^+\|_F^2 \\ \text{subject to} & (B_{k+1}^+)_{i,j} = 0 \text{ for } i \neq j \\ & (B_{k+1}^+)_{i,i} \geq \mu, \ i = 1, 2, \dots, n, \end{cases} \tag{1}$$

for some $\mu > 0$. This minimization problem has a solution

$$\left(B_{k+1}^+\right)_{i,i} = \begin{cases} b_i/Q_{i,i}, & \text{if } b_i/Q_{i,i} > \mu \\ \mu, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{b} = 2\sum_{i=\hat{1}}^{\hat{m}_c} \text{diag}(\boldsymbol{s}_i)\boldsymbol{u}_i$ and $Q = 2\sum_{i=\hat{1}}^{\hat{m}_c} [\text{diag}(\boldsymbol{s}_i)]^2$ with $\boldsymbol{s}_i \in S_k^+$ and $\boldsymbol{u}_i \in U_k^+$. In our computations, we use the inverse of this matrix, that is, $D_k^+ = \left(B_k^+\right)^{-1}$. We call this approximation D_k^+ the "*convex approximation*". The diagonal components of D_k^+ are given by

$$\left(D_{k+1}^+\right)_{i,i} = \begin{cases} \mu_{\texttt{min}}, & \text{if } Q_{i,i}/b_i < \mu_{\texttt{min}} \\ Q_{i,i}/b_i, & \text{if } Q_{i,i}/b_i > \mu_{\texttt{min}} \text{ and } Q_{i,i}/b_i < \mu_{\texttt{max}} \\ \mu_{\texttt{max}}, & \text{otherwise.} \end{cases}$$

Note that in addition to the upper bound $\mu_{\texttt{max}} = \frac{1}{\mu}$, we also use the lower bound $\mu_{\texttt{min}}$ ($0 < \mu_{\texttt{min}} < \mu_{\texttt{max}}$) for the components of the matrix. The computation of the "*concave approximation*" D_k^- is analogous.

Direction Finding, Serious and Null Steps. The SMDB uses the above mentioned diagonal approximations to compute the search direction. If the linearization error is nonnegative or if the previous step was a serious step, we use directly the "convex approximation" of the Hessian. That is, the search direction is computed by the formula

$$\boldsymbol{d}_k = -D_k^+ \tilde{\boldsymbol{\xi}}_k, \tag{2}$$

where $\tilde{\boldsymbol{\xi}}_k$ is an aggregate subgradient of the objective (to be described later). In the case of negative linearization error, we first compute the convex combination of the "convex and concave approximations" such that the combination still remains positive definite and then use this combination to compute the search direction. In other words, we compute the smallest $p_k \in [0,1]$ such that $p_k D_k^+ + (1-p_k)D_k^-$ is positive definite. Note that, being the matrices involved both diagonal, this value is very easy to compute. The search direction is then computed by the formula

$$\boldsymbol{d}_k = -(p_k D_k^+ + (1-p_k)D_k^-)\tilde{\boldsymbol{\xi}}_k. \tag{3}$$

When the search direction is computed, we next compute a new auxiliary point: $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$. A necessary condition for a *serious step* to be taken is to have

$$f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L w_k, \tag{4}$$

where $\varepsilon_L \in (0, 1/2)$ is a given descent parameter and $w_k > 0$, which will be formally defined later (Step 3 of Algorithm 3.1), represents the desirable amount of descent of f at $\boldsymbol{x}_k$. If the condition (4) is satisfied, we set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ and a serious step is taken. Note that in the case of a serious step we consider the current "convex approximation" to be good enough and we continue with this metric even if the linearization error is negative.

If condition (4) is not satisfied, we first set $t = 1$ and check if $\boldsymbol{\xi}_{k+1}^t \in \partial f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$ satisfies the null step condition

$$-\beta_{k+1}^t + \boldsymbol{d}_k^\top \boldsymbol{\xi}_{k+1}^t \geq -\varepsilon_R w_k, \tag{5}$$

where $\varepsilon_R \in (\varepsilon_L, 1)$ is a given parameter and β_{k+1}^t is the subgradient locality measure [34, 37] similar to bundle methods. That is,

$$\beta_{k+1}^t = \max\left\{|f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + t\boldsymbol{d}_k) + t(\boldsymbol{\xi}_{k+1}^t)^\top \boldsymbol{d}_k|, \gamma\|t\boldsymbol{d}_k\|^2\right\}, \tag{6}$$

where $\gamma \geq 0$ is a distance measure parameter supplied by the user. However, if condition (5) does not hold for $t = 1$, then we search for a step size $t \in (0,1)$ such that either we have a descent condition

$$f(\boldsymbol{x}_k + t\boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) - \varepsilon_L t w_k \tag{7}$$

fulfilled or $\boldsymbol{\xi}_{k+1}^t \in \partial f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$ satisfies the null step condition (5). In practice, this step size can be determined by using a line search procedure similar to the one intoroduced in [42]. Whenever the null step condition holds we perform a null step by setting $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, but information about the objective function is increased because we utilize the auxiliary point $\boldsymbol{y}_{k+1}^t = \boldsymbol{x}_k + t\boldsymbol{d}_k$ and the corresponding auxiliary subgradient $\boldsymbol{\xi}_{k+1}^t \in \partial f(\boldsymbol{y}_{k+1}^t)$

in the computation of the next aggregate value. On the other hand, the fulfilment of condition (7) during the line search leads to a serious step with $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t\boldsymbol{d}_k$ as the new iteration point.

Remark 1 Whenever the condition (4) does not hold, we start with testing if we can take a null step straight away without a line search. However, only the usage of the line search guarantees that, eventually, we either find a serious step with condition (7) or a null step with condition (5) occurs (see [42]). Nevertheless, in our numerical experiments condition (5) was always satisfied with the value $t = 1$ and no line search was needed.

For simplicity of the presentation we drop out the index t from $\boldsymbol{y}_k^t$, $\boldsymbol{\xi}_k^t$, and β_k^t even if $t \neq 1$, unless needed for clarity.

To ensure the global convergence of the SMDB, we have to assume that the sequences of matrices (D_k^+) and (D_k^-) are bounded. Due to the diagonal update formula this assumption is trivially satisfied. In addition, the condition

$$\tilde{\boldsymbol{\xi}}_k^\top D_k^+ \tilde{\boldsymbol{\xi}}_k \leq \tilde{\boldsymbol{\xi}}_k^\top D_{k-1}^+ \tilde{\boldsymbol{\xi}}_k \tag{8}$$

has to be satisfied each time there occurs more than one consecutive null step. In the SMDB this is guaranteed simply by *skipping the convex updates* if more than one consecutive null step occurs.

Aggregation. The aggregation procedure used in the SMDB is quite similar to that of the original LMBM [21, 22]. That is, we determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function

$$\begin{aligned}\varphi(\lambda_1, \lambda_2, \lambda_3) = \; & (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k)^\top D_k^+ (\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k) \\ & + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k),\end{aligned} \tag{9}$$

where m is the index after the latest serious step, and we set

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad \text{and} \tag{10}$$

$$\tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{11}$$

Algorithm. Now we give the detailed description of the SMDB.

ALGORITHM 3.1

Data: Select the positive line search parameters $\varepsilon_L \in (0, 1/2)$ and $\varepsilon_R \in (\varepsilon_L, 1)$, the initial step size $t_I \in [0.5, 1)$, and the distance measure parameter $\gamma \geq 0$ (with $\gamma = 0$ if f is convex). Choose the final accuracy tolerance $\varepsilon > 0$, the safeguard parameters $\mu_{\texttt{max}} > \mu_{\texttt{min}} > 0$, and the number of stored corrections $\hat{m}_c \geq 1$.

Step 0: (*Initialization*) Choose a starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Set $D_1^+ = I$, $\alpha_1 = 0$, $\beta_1 = 0$ and $\boldsymbol{y}_1 = \boldsymbol{x}_1$. Compute $f(\boldsymbol{x}_1)$ 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 an index for the serious step $m = k$.

Step 2: (*Convex Direction*) Compute

$$\boldsymbol{d}_k = -D_k^+ \tilde{\boldsymbol{\xi}}_k.$$

Step 3: (*Stopping Criterion*) Calculate $w_k = \tilde{\boldsymbol{\xi}}_k^\top D_k^+ \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k$. If $w_k < \varepsilon$, then stop with $\boldsymbol{x}_k$ as the final solution.

Step 4: (*Auxiliary Step*) Evaluate

$$\begin{aligned} &\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k, \\ &\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1}), \text{ and} \\ &\alpha_{k+1} = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) + \boldsymbol{\xi}_{k+1}^\top \boldsymbol{d}_k. \end{aligned}$$

Set $\boldsymbol{u}_k = \boldsymbol{\xi}_{k+1} - \boldsymbol{\xi}_m$ and $\boldsymbol{s}_k = \boldsymbol{d}_k$. If $\alpha_{k+1} \geq 0$ append these values to U_k^+ and S_k^+, respectively. Otherwise, append them to S_k^- and U_k^- .

Step 5 (*Serious Step without Line Search*) If

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

compute D_{k+1}^+ using S_k^+ and U_k^+, set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$, $f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{y}_{k+1})$, $\alpha_{k+1} = 0$, $\beta_{k+1} = 0$, $k = k+1$, and go to Step 1.

Step 6 (*Null Step Test without Line Search*) Set $t = 1$ and compute β_{k+1} as in (6). If

$$-\beta_{k+1} + \boldsymbol{d}^\top \boldsymbol{\xi}_{k+1} \geq -\varepsilon_R w_k,$$

go to Step 8.

Step 7: (*Line Search*) Determine the step size $t \in (0, t_I]$ to take either a serious step or a null step (i.e., find $t \in (0, t_I]$ for which either the condition (5) or (7) is valid). Set the corresponding values

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

In the case of the serious step, compute D_{k+1}^+ using S_k^+ and U_k^+, set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$, $f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{y}_{k+1})$, $\alpha_{k+1} = 0$, $\beta_{k+1} = 0$, $k = k+1$, and go to Step 1. Otherwise, compute β_{k+1} as in (6).

Step 8: (*Aggregation*) Determine multipliers $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function $\varphi(\lambda_1, \lambda_2, \lambda_3)$ given in (9). Set

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

Step 9: (*Null Step*) Three cases can occur:

Step 9a: $\alpha_{k+1} \geq 0$ and $m = k$ (*The First Convex Null Step*) Compute D_{k+1}^+ using S_k^+ and U_k^+. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $k = k+1$ and go to Step 2.

Step 9b: $\alpha_{k+1} \geq 0$ and $m < k$ (*The Consecutive Convex Null Step*) Set $D_{k+1}^+ = D_k^+$, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, and $k = k+1$ and go to Step 2.

Step 9c: $\alpha_{k+1} < 0$ (*Concave Null Step*) Compute D_{k+1}^- using S_k^- and U_k^- and set $D_{k+1}^+ = D_k^+$. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $k = k+1$ and go to Step 10.

Step 10: (*Concave Direction*) Compute the smallest $p \in (0, 1)$ such that the matrix $pD_k^+ +$

$(1-p)D_k^-$ remains positive definite. Compute

$$\boldsymbol{d}_k = -\left(pD_k^+ + (1-p)D_k^-\right)\tilde{\boldsymbol{\xi}}_k$$

and go to Step 3.

4. Global Convergence

We now study the convergence properties of the SMDB algorithm. First, we give the assumptions needed.

ASSUMPTION 1 The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is semismooth (see e.g. [8]), which ensures

$$f'(x,d) = \lim_{t \downarrow 0} g(x+td)^\top d,$$

with $g(x+td) \in \partial f(x+td)$.

ASSUMPTION 2 The level set $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ is bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$.

Remark 2 Semismoothness assumption guarantees that Step 7 is well posed, that is for small values of t one of the two conditions (5) and (7) is satisfied.

The optimality condition $\mathbf{0} \in \partial f(\boldsymbol{x})$ is sufficient if f is convex. However, the function f is not supposed to be convex, thus, we can only prove that the SMDB either terminates at a stationary point or generates an infinite sequence $(\boldsymbol{x}_k)$ for which accumulation points are stationary for f. In order to do this, we assume that the final accuracy tolerance ε is equal to zero. As before, we drop out the index t from $\boldsymbol{y}_k^t$, $\boldsymbol{\xi}_k^t$ and β_k^t even if $t \neq 1$.

Remark 3 The sequence $(\boldsymbol{x}_k)$ generated by Algorithm 3.1 is bounded by Assumption 2 and the monotonicity of the sequence (f_k). The monotonicity of (f_k) is guaranteed since either the condition (4) or (7) is satisfied for serious steps, while $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps.

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 [13]. The matrix D^+ (D^-) is bounded due to the fact that all its components are in the closed interval $[\mu_{\texttt{min}}, \mu_{\texttt{max}}]$ ($[-\mu_{\texttt{max}}, -\mu_{\texttt{min}}]$). Thus, the set of the search directions $\boldsymbol{d}_k$ and the sequence $\boldsymbol{y}_k$ are also bounded.

We start the convergence analysis by giving three technical results (Lemmas 4.1, 4.2, and 4.3). After that, we prove (in Theorem 4.4) that having the value $w_k = 0$ implies that the corresponding point $\boldsymbol{x}_k$ is a stationary point for the objective function. For an infinite sequence $(\boldsymbol{x}_k)$, we first show (in Lemma 4.5) that if $(\boldsymbol{x}_k)_{k\in\mathcal{K}} \to \bar{\boldsymbol{x}}$ and $(w_k)_{k\in\mathcal{K}} \to 0$ for some subset $\mathcal{K} \subset \{1, 2, \ldots\}$, then the accumulation point $\bar{\boldsymbol{x}}$ is a stationary point for the objective function. Furthermore, using the fact that the sequence (w_k) is nonincreasing in the consecutive null steps (see Lemma 4.6), we prove that the indefinite sequence of consecutive null steps with $\boldsymbol{x}_k = \boldsymbol{x}_m$ implies $\mathbf{0} \in \partial f(\boldsymbol{x}_m)$ (in Lemma 4.7). Finally, in Theorem 4.8 we combine all the results obtained and show that every accumulation point of $(\boldsymbol{x}_k)$ is stationary for the objective function.

LEMMA 4.1 *At the kth iteration of Algorithm 3.1, we have*

$$w_k = \tilde{\boldsymbol{\xi}}_k^\top D_k^+ \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \mu_{\texttt{min}} \|\tilde{\boldsymbol{\xi}}_k\|^2,$$

and

$$\beta_{k+1} \geq \gamma \|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\|^2. \tag{12}$$

Proof. We point out first that $\tilde{\beta}_k \geq 0$ for all k by equations (6), (11), and Step 1 in Algorithm 3.1. The relations

$$w_k = \tilde{\boldsymbol{\xi}}_k^\top D_k^+ \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k, \qquad w_k \geq 2\tilde{\beta}_k, \qquad w_k \geq \mu_{\texttt{min}} \|\tilde{\boldsymbol{\xi}}_k\|^2$$

follow immediately from (1), Step 3 in Algorithm 3.1 and the lower bound $\mu_{\texttt{min}}$ used for the matrices.

By (6) and since we have $\beta_{k+1} = 0$ and $\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}\| = 0$ for serious steps and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ for null steps, condition (12) always holds for some $\gamma \geq 0$.

■

LEMMA 4.2 *Suppose that Algorithm 3.1 is not terminated before the kth iteration. Then, there exist numbers $\lambda^{k,j} \geq 0$ for $j = 1, \ldots, k$ and $\tilde{\sigma}_k \geq 0$ such that*

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

Proof. See the proof of Lemma 3.2 in [42]. ■

LEMMA 4.3 *Let $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ be given and suppose that there exist vectors $\bar{\boldsymbol{g}}$, $\bar{\boldsymbol{\xi}}_i$, $\bar{\boldsymbol{y}}_i$, and numbers $\bar{\lambda}_i \geq 0$ for $i = 1, \ldots, l$, $l \geq 1$, such that*

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

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

Proof. See the proof of Lemma 3.3 in [42]. ■

In the following theorem we assume $\epsilon = 0$.

THEOREM 4.4 *If Algorithm 3.1 terminates at the kth iteration, then the point $\boldsymbol{x}_k$ is stationary for f.*

Proof. If Algorithm 3.1 terminates at Step 3, then the fact $\varepsilon = 0$ implies that $w_k = 0$. Thus, $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{0}$ and $\tilde{\beta}_k = \tilde{\sigma}_k = 0$ by Lemma 4.1 and Lemma 4.2.

Now, by Lemma 4.2 and by using Lemma 4.3 with

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

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

From now on, we suppose that Algorithm 3.1 does not terminate, that is, $w_k > 0$ for all k.

LEMMA 4.5 *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded. If there exist a point* $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ *and an infinite set* $\mathcal{K} \subset \{1, 2, \ldots\}$ *such that* $(\boldsymbol{x}_k)_{k\in\mathcal{K}} \to \bar{\boldsymbol{x}}$ *and* $(w_k)_{k\in\mathcal{K}} \to 0$, *then* $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$.

Proof. The proof is similar to the proof of Lemma 3.4 in [42]. ■

LEMMA 4.6 *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded, the number of serious steps is finite, and the last serious step occurred at the iteration* $m-1$. *Then*

$$\tilde{\boldsymbol{\xi}}_{k+1}^\top D_{k+1}^+ \tilde{\boldsymbol{\xi}}_{k+1} = \tilde{\boldsymbol{\xi}}_{k+1}^\top D_k^+ \tilde{\boldsymbol{\xi}}_{k+1} \qquad \textit{and} \tag{13}$$

$$\operatorname{tr}(D_k^+) \leq \mu_{\texttt{max}} n \tag{14}$$

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

Proof. For all $k > m$ we have $D_{k+1}^+ = D_k^+$ due to fact that we use either Step 9b or Step 9c of Algorithm 3.1 at null steps. Thus, condition (13) is valid.

Furthermore, we have

$$\begin{aligned} \operatorname{tr}(D_k^+) - \mu_{\texttt{max}} n &= \operatorname{tr}(D_k^+) - \mu_{\texttt{max}} \operatorname{tr}(I) \\ &= \operatorname{tr}(D_k^+) - \operatorname{tr}(\mu_{\texttt{max}} I) \\ &= \operatorname{tr}(D_k^+ - \mu_{\texttt{max}} I) \\ &\leq 0 \end{aligned}$$

for all k, since D_k^+ is a diagonal matrix with the largest diagonal element less than or equal to $\mu_{\texttt{max}}$. Therefore, condition (14) is valid for all $k > m$. ■

LEMMA 4.7 *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded, the number of serious steps is finite, and the last serious step occurred at the iteration* $m-1$. *Then, the point* $\boldsymbol{x}_m$ *is stationary for* f.

Proof. From (9), (10), (11), Lemma 4.1, and Lemma 4.6 we obtain

$$\begin{aligned} w_{k+1} &= \tilde{\boldsymbol{\xi}}_{k+1}^\top D_{k+1}^+ \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &= \tilde{\boldsymbol{\xi}}_{k+1}^\top D_k^+ \tilde{\boldsymbol{\xi}}_{k+1} + 2\tilde{\beta}_{k+1} \\ &= \varphi(\lambda_1^k, \lambda_2^k, \lambda_3^k) \\ &\leq \varphi(0,0,1) \\ &= \tilde{\boldsymbol{\xi}}_k^\top D_k^+ \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k \\ &= w_k \end{aligned} \tag{15}$$

for $k > m$.

Let us denote $D_k^+ = W_k^\top W_k$. Then, the function φ (see (9)) can be given in the form

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

From (15) we obtain the boundedness of the sequences (w_k), $(W_k\tilde{\boldsymbol{\xi}}_k)$, and $(\tilde{\beta}_k)$. Furthermore, Lemma 4.6 and Remark 3 assures the boundedness of (D_k), (W_k), $(\boldsymbol{y}_k)$, $(\boldsymbol{\xi}_k)$, and $(W_k\boldsymbol{\xi}_{k+1})$.

Now, the last part of the proof proceeds similarly to the proof (part (ii)) of Lemma 3.6 in [42]. ■

THEOREM 4.8 *Suppose that the level set* $\{\, \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \,\}$ *is bounded. Then, every accumulation point of the sequence* $(\boldsymbol{x}_k)$ *is stationary for* f.

Proof. Let $\bar{\boldsymbol{x}}$ be an accumulation point of $(\boldsymbol{x}_k)$, and let $\mathcal{K} \subset \{1, 2, \ldots\}$ be an infinite set such that $(\boldsymbol{x}_k)_{k\in\mathcal{K}} \to \bar{\boldsymbol{x}}$. In view of Lemma 4.7, we can restrict our consideration to the case where the number of serious steps is infinite. We denote

$$\begin{aligned} \mathcal{K}' = \{\, k \mid \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t\boldsymbol{d}_k \text{ with } t > 0 \text{ and} \\ \text{there exists } i \in \mathcal{K},\ i \leq k \text{ such that } \boldsymbol{x}_i = \boldsymbol{x}_k \}. \end{aligned}$$

Obviously, $\mathcal{K}'$ is infinite and $(\boldsymbol{x}_k)_{k\in\mathcal{K}'} \to \bar{\boldsymbol{x}}$. The continuity of f implies that $(f(\boldsymbol{x}_k))_{k\in\mathcal{K}'} \to f(\bar{\boldsymbol{x}})$, thus, $f(\boldsymbol{x}_k) \downarrow f(\bar{\boldsymbol{x}})$ by the monotonicity of the sequence $(f(\boldsymbol{x}_k))$ obtained due to the descent step conditions (4) and (7). Using conditions (4) and (7) and the fact that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ in null steps, we obtain

$$0 \leq t\varepsilon_L w_k \leq f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \to 0 \qquad \text{for } k \geq 1. \tag{16}$$

Thus, if the set $\mathcal{K}_1 = \{\, k \in \mathcal{K}' \mid t \geq t_{\texttt{min}} \}$ is infinite for some bound $t_{\texttt{min}} > 0$ then $(w_k)_{k\in\mathcal{K}'} \to 0$ and $(\boldsymbol{x}_k)_{k\in\mathcal{K}'} \to \bar{\boldsymbol{x}}$ by (16). Hence, by Lemma 4.5 we have $\mathbf{0} \in \partial f(\bar{\boldsymbol{x}})$.

In the other case, where the set $\mathcal{K}_1$ is finite, the result is obtained the same way as in the proof of Theorem 3.2 in [42]. ■

Note that, if we choose $\varepsilon > 0$, Algorithm 3.1 terminates in a finite number of steps.

5. Numerical Experiments

In this section, we compare the SMDB with the LMBM and the D-BUNDLE. The test set used in our experiments consists of large-scale nonsmooth minimization problems first introduced in [21]. These problems can be formulated with any number of variables. We have tested these problems with 1000, 10000, 100000 and one million variables. Problems 1 – 5 are convex while problems 6 – 10 are nonconvex. We ran each problem 10 times: first with fixed starting point $\bar{\boldsymbol{x}}_1$ given in [21] and then the remaining nine times using a starting point generated randomly from a ball centered at $\bar{\boldsymbol{x}}_1$ with a radius $\|\bar{\boldsymbol{x}}_1\|/n$. Other numerical experiments comparing different NSO solvers including the LMBM and the D-BUNDLE can be found for instance in [5, 28, 29].

Solvers and Parameters. We now give a brief description of each software, the parameters used, and the references from which the code can be downloaded.

`LMBM` is an implementation of the LMBM [21, 22] specifically developed for large-scale NSO. The solver uses the modified weak Wolfe-type line search for finding suitable step sizes (see, [22]). 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 *seven* and the maximum number of stored correction pairs equal to 15. Moreover, the maximum size of the bundle was set to *two* since previous tests have shown that with extremely large problems this is the only practical option (see [28]). Otherwise, the default parameters of the code were used.

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

`D-Bundle` is a diagonal bundle solver developed specially for sparse nonsmooth minimization [28]. Similarly to `LMBM`, the solver uses the modified weak Wolfe-type line search to find suitable step sizes (see, [22]). With `D-Bundle` we used the maximum number of stored correction pairs equal to seven. In addition, the maximum size of the bundle was set to *two*. For all other parameters we have used the default settings of the code.

The Fortran 95 source code is available for downloading from http://napsu.karmitsa.fi/dbundle/.

`SMDB` is an implementation of the splitting metrics diagonal bundle method introduced in this paper. The parameters used with `SMDB` are

$$\mu_{\texttt{min}} = 10^{-10}, \qquad \mu_{\texttt{max}} = 1.0,$$
$$\varepsilon_L = 10^{-4}, \qquad \varepsilon_R = 0.25,$$

and

$$\gamma = \begin{cases} 0.0 & \text{for convex } f, \\ 10^{-4} & \text{for nonconvex } f. \end{cases}$$

Similarly to `D-Bundle` the number of stored correction pairs was set to *seven* and the maximum size of the bundle was set to *two*.

As already mentioned the line search was never needed with `SMDB` but the step size $t = 1$ was always accepted either as a serious or null step. In addition to the basic code

SMDB, we implemented the codes using the Armijo-type line search and the nonmonotonic Armijo-type line search. We denote these codes by SMDBA and SMDBNM, respectively. The idea here is to seek for a suitable step size such that a serious step would occur. That is, we perform at most some predetermined number of line searches and we check condition (7) (or the modified serious step condition (17) given below) and only if none of the step sizes gives us a serious step we do a null step. Note that no theoretical guarantee of the satisfaction of the null step condition (5) after these Armijo-type line searches is given but, in our numerical experiments it was always satisfied.

The parameters used with SMDBA and SMDBNM are the same as with SMDB. With SMDBA the maximum number of Armijo search was set to *two*. With SMDBNM we used at most ten previous function values obtained at serious steps to test the modified serious step condition

$$f(\boldsymbol{y}_{k+1}) \leq \max_{0 \leq j \leq m(k)} f(\boldsymbol{x}_{i-j}) - \varepsilon_L w_k, \tag{17}$$

where $i \in K = \{l \mid \boldsymbol{x}_{l+1} = \boldsymbol{x}_l + t_l \boldsymbol{d}_l\}$ and $m(k) = \min\{m(k-1)+1, 10\}$ if $i = k$ and $m(k) = \min\{m(k-1), 10\}$ otherwise (i.e. at null steps). The maximum number of Armijo searches was set to 20.

The Fortran 95 source code of SMDB is available for downloading from http://napsu.karmitsa.fi/smdb/ and it includes the codes SMDBA and SMDBNM.

The experiments were performed on an Intel® Core™ i5, 1.60GHz. To compile the codes, we used gfortran, the GNU Fortran compiler.

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

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

where $f_{\texttt{best}}$ is a solution obtained with the solver and $f_{\texttt{opt}}$ 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*. In addition to the usual stopping criteria of the solvers, we terminated the experiments if the elapsed CPU time exceeded two hours.

Results. The results are summarized in Tables 1 – 4 and in Figures 2 – 9. 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). In Tables 1 – 4, the results are given for problems with the original starting points given in [21]. These results show quite well the overall performance of the solvers. We have used bold-face text to emphasize the best results. An asterix after a result means that the result obtained was inaccurate. Note that with the nonconvex problems it was not always easy to say whether the solution obtained was a local solution (or a stationary point) or not. Thus, we have only accepted the results that converge to the global minimum. In addition, the results are analyzed using the performance profiles (Figures 2 – 9) introduced in [16]. In Figures 2 – 9 we first give the results for all problems with given n (figures a). Then we separate the convex (figures b) from the nonconvex (figures c) problems. In 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; that is, it uses least computational time or evaluations.

On the other hand, the value of $\rho_s(\tau)$ at the rightmost abscissa gives the percentage of test problems that the corresponding solver can solve; in other words, the reliability of the solver. In addition, the relative efficiency of each solver can be directly seen from the performance profiles: the higher the particular curve, the better the corresponding solver. In performance profiles inaccurate results were considered as failures.

Table 1. Summary of the results with 1000 variables.

P	LMBM nfg/cpu	D-Bundle nfg/cpu	SMDB nfg/cpu	SMDBA nfg/cpu	SMDBNS nfg/cpu
1	37 728/0.97*	6 136/0.09	63 858/1.69	**3 002**/0.07	5999/**0.06**
2	fail	fail	fail	fail	fail
3	3 292/0.03	3 751/0.03	61 436/2.87	**222**/**0.01**	918/0.02
4	3 450/0.04	6 917/0.08	81 449/4.68	**240**/**0.01**	627/0.02
5	326/**0.01**	1 388/**0.01**	fail	**223**/**0.01**	719/0.03
6	1 138/**0.01**	1 075/0.02	1 719/0.06	**710**/0.02	983/0.04
7	5 690/0.45	13 319/0.84	**246**/**0.04**	663/0.05	1225/0.14
8	**6 020**/**0.05**	7 617/**0.05**	1 000 166/42.78	1 999 991/55.89	192 513/2.74
9	1 128/0.01	1 106/0.01	fail	**112**/**0.00**	fail
10	11 282/0.10	17 377/0.11	454/0.02	**338**/**0.01**	1 302/0.04

With 1000 variables, the new solver SMDBA with few rounds of Armijo line search was clearly the best when compared with the other methods (see Table 1 and Figures 2 and 6). In nonconvex settings (see Figures 2(c) and 6(c)), we can say that it was superior: it was the most efficient method in 40% of the problems and succeeded in solving 94% of them. In addition, the overall performance of SMDBNM with nonmonotone line search was very good: in convex problems it was the most reliable solver, although, D-Bundle did not fall far behind (see Figures 2(b) and 6(b)).

While SMDBA and SMDBNM seem to be efficient both in convex and nonconvex setting, the solver SMDB without the line search solved more efficiently the nonconvex problems using quite a large number of evaluations in the convex cases. On the other hand, D-Bundle was clearly more reliable in convex settings.

With 10 000 variables, the superiority of SMDBA is not at all clear anymore (see Table 2 and Figures 3 and 7). In fact, here all the other solvers but D-Bundle were usually both more efficient and more reliable than SMDBA when only nonconvex problems are considered (see Figures 3(c) and 7(c)). Moreover, the overall performance of SMDBA was not very convincing in the convex case, either (see Figures 3(b) and 7(b)). In convex problems D-Bundle and SMDBNM seem to be the best performing solvers while SMDB again used lots of evaluations and failed in solving more than 50% of these problems. However, in nonconvex settings SMDB was among the most reliable solvers together with SMDBNM and LMBM. Here, SMDBNM was also the most efficient solver.

A problem with 100 000 variables can be considered as an extremely large nonsmooth problem. Here, D-Bundle was the most efficient solver both in convex (44%) and in nonconvex (30%) settings (see Table 3, and Figures 4 and 8). Nevertheless, the overall performances of LMBM, D-Bundle and SMDNNM were quite similar.

None of the solvers succeeded in solving more than 65% of problems with 100 000 variables with the desired accuracy. In convex case (see Problems 1–5 in Table 3 and Figures 4(b) and 8(b)) all the solvers but SMDBNM failed to solve Problems 1 and 2 from all starting points. SMDBNM solved Problem 1 in two cases out of ten but also SMDBNM always failed in solving Problem 2. We will analyse these failures later. With nonconvex problems (see Problems 6–10 in Table 3 and Figures 4(c) and 8(c)) there was quite a big variation in the performance of the solvers with different problems, there was no

Table 2. Summary of the results with 10 000 variables.

P	LMBM nfg/cpu	D-Bundle nfg/cpu	SMDB nfg/cpu	SMDBA nfg/cpu	SMDBNM nfg/cpu
1	fail	179 502/26.41	913 677/242.30	**30 014/5.92**	60003/6.40
2	fail	fail	fail	fail	fail
3	6 082/0.52	3 669/0.27	31 482/12.89	**311/0.08**	1 059/0.30
4	7 080/0.81	**5 144/0.58**	1 000 104/515.11	528 300/206.16	9 007/2.68
5	**244/0.04**	307/**0.04**	fail	5 643/1.99	792/0.27
6	10 108/1.55	10 104/2.03	15 197/4.42	6 421/1.64	**1 175/0.31**
7	8 888/6.26	49 823/32.95	**360/0.51**	1 186/1.47	895/1.14
8	**6 232/0.61**	11 261/0.88	1 000 182/372.28	1 999 992/526.40	361 347/48.87
9	fail	474/0.06*	fail	fail	fail
10	fail	fail	3 232/1.44	113 090/37.47	**1 197/0.39**

systematic failure in any of the problems, and we can not say that one solver was clearly better than another, although, quite surprisingly, D-Bundle was the most reliable method tested in nonconvex settings followed by SMDBNM and SMDB. In addition, as said before, with the nonconvex problems it is not always easy to say whether the solution obtained is a local solution (or a stationary point) or not.

Table 3. Summary of the results with 100 000 variables.

P	LMBM nfg/cpu	D-Bundle nfg/cpu	SMDB nfg/cpu	SMDBA nfg/cpu	SMDBNM nfg/cpu
1	fail	fail	fail	fail	**396 980/578.91**
2	fail	fail	fail	fail	fail
3	5 796/5.59	**144/0.13**	1 000 106/4 598.92	5 145/13.91	1 270/3.58
4	10 424/12.77	**584/0.82**	1 000 062/5 499.51	1 877 517/7 200.00	5 125/16.35
5	**438**/1.09	816/**0.84**	fail	8 475/29.15	772/2.71
6	100 142/**166.46**	**100 100**/237.49	fail	fail	fail
7	fail	53 905/339.49	**310/5.00**	7 188/91.92	2 251/25.18
8	**2 100/4.05**	5 141/5.18	1 000 131/4 129.34	1 999 993/5033.66	779 804/1 059.88
9	**1 400/3.87**	1 086/2.30*	fail	fail	fail
10	34 630/34.55*	fail	14 755/69.18	5 034/15.81	**1 757/5.78**

With one million variables SMDBNM succeed in solving 60 % of problems, hence, it was the most robust of the solvers (see Table 4 and Figures 5 and 9). All the other solvers succeed in solving about 50 % problems, although, the solvers usually converged to the same (stationary?) point also in Problem 6. One could say that solving only about 50 or 60 percentages of problems is not very convincing, but it is better than most nonsmooth algorithms can do [5, 28, 29]. With convex problems, D-Bundle was superior in efficiency and it was also the most robust solver together with LMBM and SMDBNM. However, with nonconvex problems it failed to solve 70% of problems. Here, the new solvers were the most robust ones.

We will now consider more closely some problems that caused biggest failures in our experiments. In Problem 1, LMBM is known to have difficulties due to sparse structure of the problem and the dense approximation of Hessian used in LMBM [21, 22]. Indeed, LMBM always failed to solve Problem 1 with the desired accuracy. All the other algorithms use only a diagonal approximation to Hessian and they did a little better: they succeeded in solving this problem with 1 000 and 10 000 variables and SMDBNM also two times with 100 000 variables. Moreover, it has been noted in [5] that the piecewise linear problem 2 is very difficult to most NSO algorithms already in rather small dimensions. The proximal bundle method [36] that uses piecewise linear model was superior in this problem (see

Table 4. Summary of the results with one million variables.

P	LMBM nfg/cpu	D-Bundle nfg/cpu	SMDB nfg/cpu	SMDBA nfg/cpu	SMDBNM nfg/cpu
1	fail	fail	fail	fail	fail
2	fail	fail	fail	fail	fail
3	1 698/25.42	**367/4.67**	169 205/7 200.07	254 334/7 200.00	103 873/1 666.90
4	14 302/216.11	28 099/349.38	129 821/7 200.06	177 304/7 200.12	**3 698/64.92**
5	1 580/45.39	**772/9.08**	fail	70 024/2 518.19	11 204/141.74
6	fail	fail	fail	fail	fail
7	fail	fail	**317/52.16**	fail	2 789/313.38
8	**2 632/93.27**	19 727/195.34	174 037/7 200.08	279 626/7 200.04	119 455/1 765.93
9	**2748/107.13**	3 569/79.64*	fail	fail	fail
10	fail	14 521/173.89*	**41 301/1 948.28**	222 599/7 200.07	349 915/7 200.22

[5]). In the current settings none of the solvers (which, on the other hand, do not have piecewise linear model as the proximal bundle method) could solve the problem. Note, however, that we use here extra large dimensions. Solver SMDB without any line search tends to take many steps (especially in convex settings) and, thus, it sometimes failed due to maximum number of evaluations used. In practice, this means that the model used is not accurate enough to provide a good search direction that can be accepted as a serious step with step length one. In addition, SMDB almost always failed to solve convex Problem 5, probably due to numerical difficulties with floating point numbers. In addition to the above mentioned problems, there were no problems where one or more of the algorithms would have systematically failed. However, Problem 8 was somewhat difficult for new solvers SMDB and SMDBA: they always needed the maximum number of iterations to solve the problem. Note that with all the solvers, some of the failures and inefficiencies could have been avoided if suitable parameters would have been chosen. However, we ran all our test cases with the same sets of parameters.

From performance profiles (see Figures Figures 2 – 9) we can conclude that the new splitting metrics procedure used in SMDB, SMDBA, and SMDBNM does give some advantage over the traditional subgradient locality measures in nonconvex cases. In addition, in convex case, already two rounds of Armijo-like line search made a big difference both in efficiency and in reliability of solvers (cf. SMDB and SMDBA), while the nonmonotonic Armijo-type line search (i.e. SMDBNM) gave even more advantage. Nevertheless, the difference between no line search and two rounds Armijo line search were opposite in nonconvex settings and SMDB was, in fact, the most robust solver together with SMDBNM.

6. Conclusions

In this paper, we have described a new splitting metrics diagonal bundle method (SMDB) 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 nor convex.

The numerical experiments were made using three different versions of the new method: one with no line search in serious steps, second with few rounds of Armijo-type line search and the third with nonmonotone Armijo-type line search. The results reported show that the new splitting metrics procedure does give some advantage over the traditional subgradient locality measures in nonconvex settings. With large nonconvex problems ($n = 1\,000$) SMDB with few rounds of Armijo line search was superior to other solvers including LMBM and D-Bundle used as benchmarks. However, with larger number of

variables it needed more function and subgradient evaluations than those other solvers. On the other hand, with one million variables SMDB with nonmonotone line search was the most robust of the solvers tested and SMDB without any line search was the most efficient together with `LMBM` on nonconvex problems.

We can conclude that SMDB is a good alternative to existing nonsmooth optimization algorithms for large scale optimization and it can be used to solve extremely large-scale nonsmooth problems.

Acknowledgments and Funding

The work was financially supported by the Academy of Finland (Project No. 289500 and 294002), Universitá della Calabria, the Jenny and Antti Wihuri Foundation and the University of Turku Graduate School UTUGS Matti Programme. A part of the work was accomplished while the corresponding author was visiting Faculty of Science and Technology, Federation University Australia.

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., Karmitsa, N., and Mäkelä, M. M. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014.

[6] Bagirov, A., Taheri, S., and Ugon, J. Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. *Pattern Recognition 53* (2016), 12–24.

[7] 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 34*, 6 (2012), 1068–1079.

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

[9] 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.

[10] 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.

[11] 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.

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

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

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

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

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

[17] Fuduli, A., Gaudioso, M., and Giallombardo, G. A DC piecewise affine model and a bundling technique in nonconvex nonsmooth minimization. *Optimization Methods and Software 19*, 1 (2004), 89–102.

[18] 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.

[19] Fuduli, A., Gaudioso, M., and Nurminski, E. A. A splitting bundle approach for non-smooth non-convex minimization. *Optimization: A Journal of Mathematical Programming and Operations Research 64*, 5 (2015), 1131–1151.

[20] Gaudioso, M., and Gorgone, E. Gradient set splitting in nonconvex nonsmooth numerical optimization. *Optimization Methods and Software 25* (2010), 59–74.

[21] 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.

[22] 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.

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

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

[25] 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).

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

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

[28] Karmitsa, N. Diagonal bundle method for nonsmooth sparse optimization. *Journal of Optimization Theory and Applications 166*, 3 (2015), 889–905. DOI 10.1007/s10957-014-0666-8.

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

[30] Karmitsa, N., Bagirov, A., and Taheri, S. Diagonal bundle method for solving the minimum sum-of-squares clustering problems. TUCS Technical Report, No. 1156, Turku Centre for Computer Science, Turku, 2016. The report is available online at http://tucs.fi/publications/view/?pub_id=tKaBaTa16a.

[31] Karmitsa, N., Bagirov, A., and Taheri, S. Mssc clustering of large data using the limited memory bundle method. TUCS Technical Report, No. 1164, Turku Centre for Computer Science, Turku, 2016. The report is available online at http://tucs.fi/publications/view/?pub_id=tKaBaTa16b.

[32] Karmitsa, N., Tanaka Filho, M., and Herskovits, J. Globally convergent cutting plane method for nonconvex nonsmooth minimization. *Journal of Optimization Theory and Applications 148*, 3 (2011), 528–549.

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

[34] 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.

[35] 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.

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

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

[38] 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.

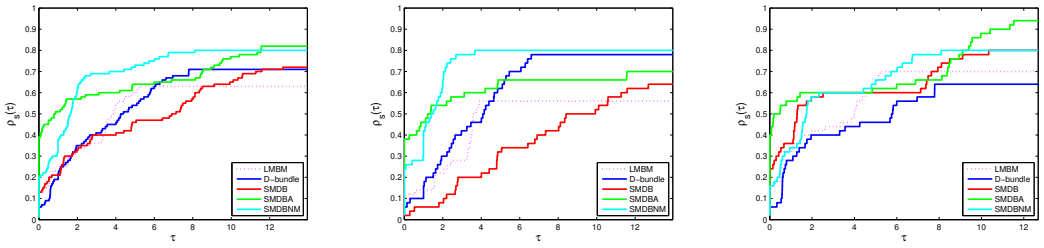
[39] Moreau, J., Panagiotopoulos, P. D., and Strang, G., Eds. *Topics in Nonsmooth Mechanics*. Birkhäuser Verlag, Basel, 1988.

[40] 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.

[41] 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.

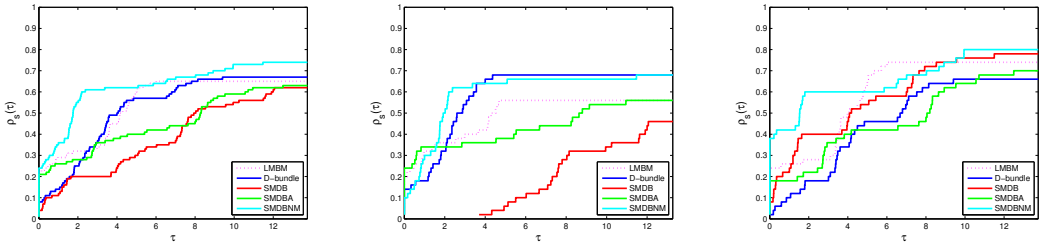
[42] Vlček, J., and Lukšan, L. Globally convergent variable metric method for nonconvex nondif-

ferentiable unconstrained minimization. *Journal of Optimization Theory and Applications 111*, 2 (2001), 407–430.



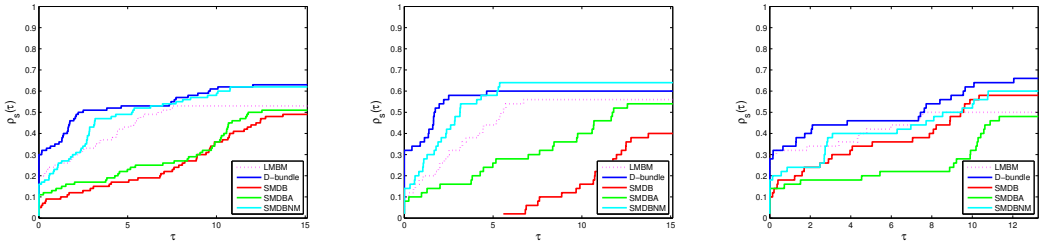
(a) All (b) Convex (c) Nonconvex

Figure 2. Evaluations with 1 000 variables.



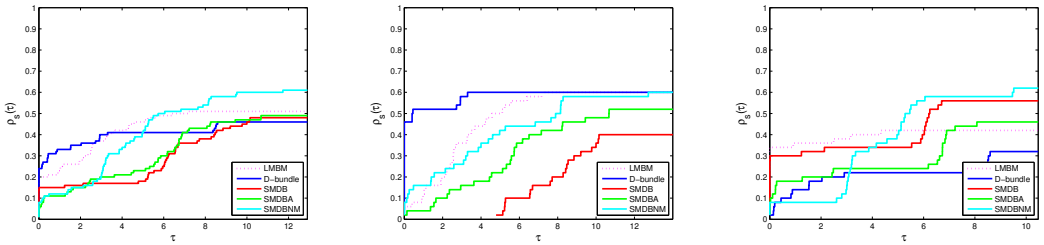
(a) All (b) Convex (c) Nonconvex

Figure 3. Evaluations with 10 000 variables.



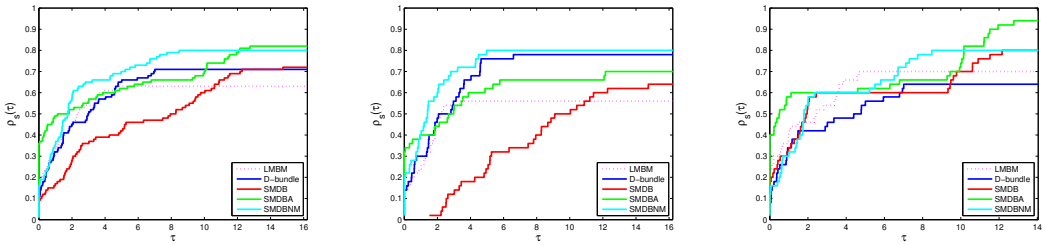
(a) All (b) Convex (c) Nonconvex

Figure 4. Evaluations with 100 000 variables.



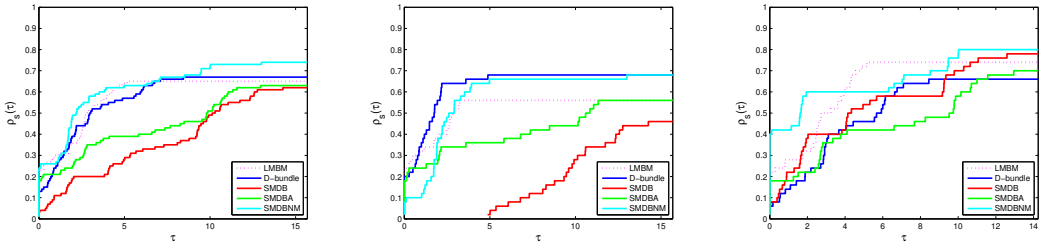
(a) All (b) Convex (c) Nonconvex

Figure 5. Evaluations with 1000 000 variables.



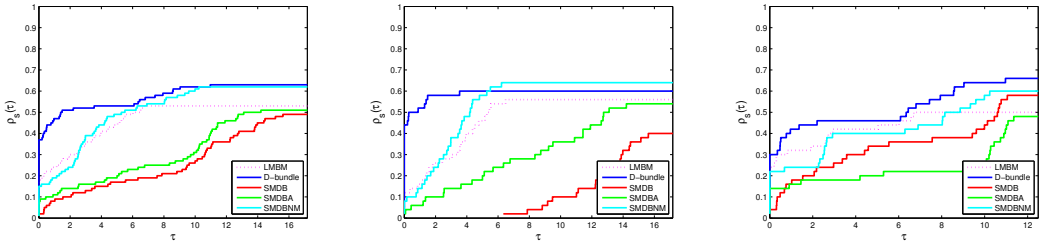
(a) All (b) Convex (c) Nonconvex

Figure 6. CPU-time with 1 000 variables.



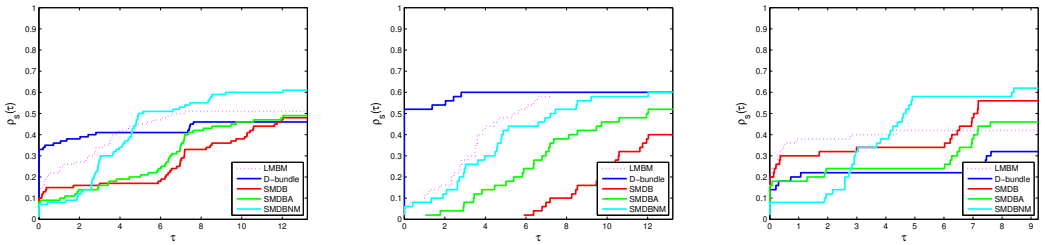
(a) All (b) Convex (c) Nonconvex

Figure 7. CPU-times with 10 000 variables.



(a) All (b) Convex (c) Nonconvex

Figure 8. CPU-times with 100 000 variables.



(a) All (b) Convex (c) Nonconvex

Figure 9. CPU-times with 1000 000 variables.