Napsu Karmitsa | Manlio Gaudioso | Kaisa Joki

# Splitting Metrics Diagonal Bundle Method for Large-Scale Nonconvex Nonsmooth Optimization

Turku Centre *for* Computer Science

TUCS Technical Report
No 1178, March 2017

# Splitting Metrics Diagonal Bundle Method for Large-Scale Nonconvex Nonsmooth Optimization

Napsu Karmitsa
Department of Mathematics and Statistics
University of Turku
FI-20014 Turku, Finland
napsu@karmitsa.fi

Manlio Gaudioso
Dipartimento di Elettronica e Sistemistica,
Universitá della Calabria,
87036, Rende (CS), Italy
gaudioso@deis.unical.it

Kaisa Joki
Department of Mathematics and Statistics
University of Turku
FI-20014 Turku, Finland
kjjoki@utu.fi

TUCS Technical Report
No 1178, March 2017

**Abstract**

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 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 new method.

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

**TUCS Laboratory**
Turku Optimization Group (TOpGroup)

# 1 Introduction

Nonsmooth optimization (NSO) refers to the general problem of minimizing (or maximizing) functions that are typically not differentiable at their minimizers (maximizers) (see e.g., [5]). NSO problems are encountered in many application areas: for instance, in economics [40], mechanics [38], engineering [37], control theory [14], optimal shape design [23], machine learning [26], and data mining [4, 9] including cluster analysis [6, 15, 29, 30] and classification [2, 3, 7, 12]. Most of these problems are large-scale and nonconvex.

In this paper, we are considering of 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 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 convexity of the problem. Fortunately, several nonconvex algorithms have been introduced only recently, for instance, in [1, 10, 16, 17, 18, 19, 21, 22, 31, 39]. Nevertheless, also most of these algorithms are developed only for small-scale problems.

The convex model of the objective function is usually reasonable good also for nonconvex problems, but 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 [16, 19] 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 of maintaining 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, [27]) 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, [20, 21]) and the *variable metric bundle method* (VMBM, [34, 41]) 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 [16, 19]. 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. [25, 32, 35]) needs not to be solved, nor the number of stored subgradients needs 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 and, 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, finally, Section 6 concludes the paper.

# 2 Notations and Background

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

We denote by $\|\cdot\|$ the Euclidean norm in $\mathbb{R}^n$ and by $\boldsymbol{a}^T\boldsymbol{b}$ 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 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 $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. 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 $k \to \infty$.

# 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 value of the objective function $f(\boldsymbol{x})$ and 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 the cases 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 details the different components of the method and then introduce the entire algorithm.

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

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

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 $\alpha_{k+1}$, respectively.

**Matrix Updating and Splitting of Data.** We use the diagonal update formula introduced in [24] 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 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 $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and thus, we use here the auxiliary point $\boldsymbol{y}_{k+1}$ instead of $\boldsymbol{x}_{k+1}$.

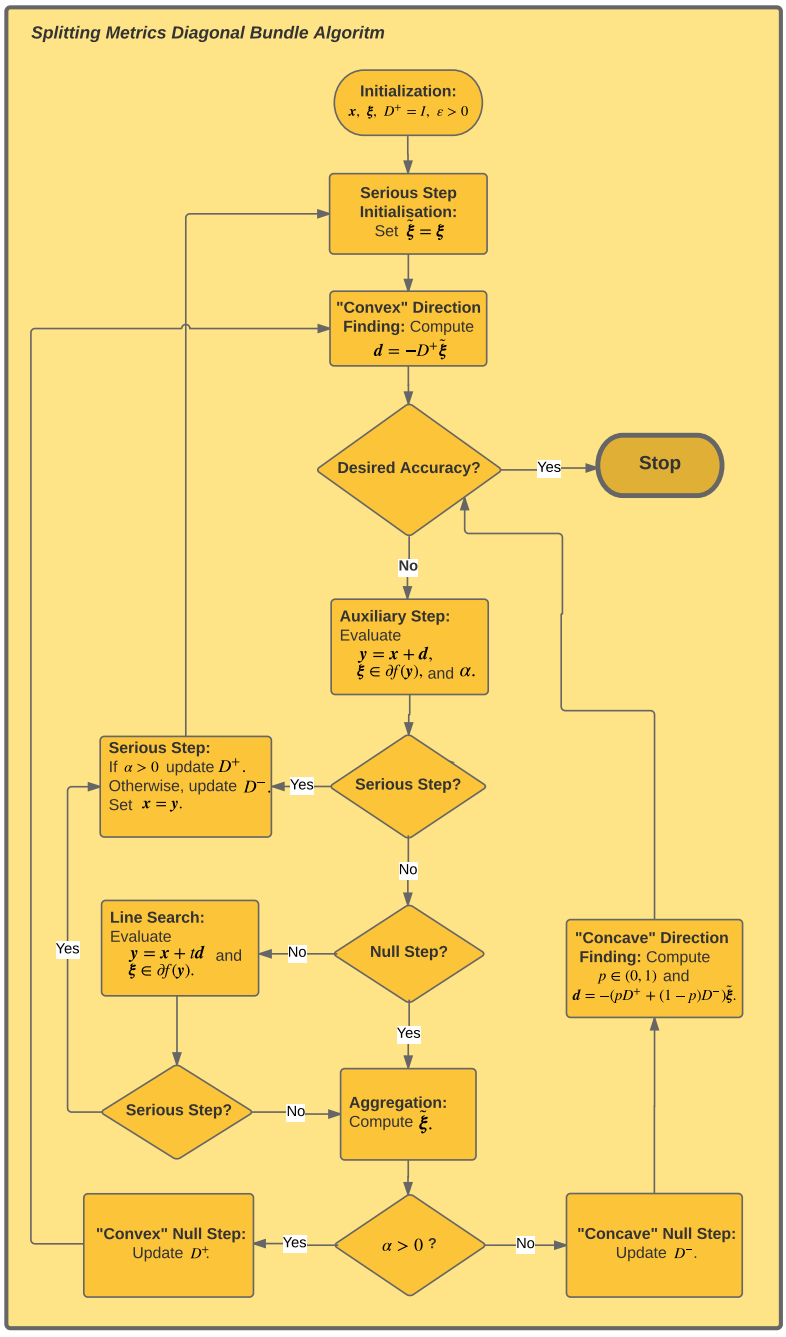


Figure 1: Flowchart of the SMDB. Here, $\tilde{\xi}$ is an aggregate subgradient, $D^+$ and $D^-$ are the convex and the concave update matrices, respectively, $\alpha$ is the linearization error, and $\varepsilon$ is the stopping tolerance.

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, also in the D-BUNDLE [27], we now use different corrections matrices depending on the sign of the linearization error. That is, we add $\boldsymbol{s}_k$ 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 both in $S_k^+$ ($U_k^+$) and $S_k^-$ ($U_k^-$). For the 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 that $m_c$ 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 $\boldsymbol{b} = 2\sum_{i=\hat{1}}^{\hat{m}_c} \operatorname{diag}(\boldsymbol{s}_i)\boldsymbol{u}_i$ and $Q = 2\sum_{i=\hat{1}}^{\hat{m}_c} [\operatorname{diag}(\boldsymbol{s}_i)]^2$ 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

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

Note that in addition to the upper bound $\mu_{max} = \frac{1}{\mu}$, we also use the lower bound $\mu_{min}$ ($0 < \mu_{min} < \mu_{max}$) 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 non negative 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 metrics even if the linearization error could be negative.

If the 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

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

where $\varepsilon_R \in (\varepsilon_L, 1/2)$ is a given parameter and $\beta_{k+1}^t$ is the subgradient locality measure [33, 36] similar to bundle methods. That is,

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

where $\gamma \geq 0$ is a distance measure parameter supplied by the user. However, if the 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 [41]. 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 next aggregate values. On the other hand, the fulfilment of the 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 3.1. Whenever the condition (4) does not hold, we start with testing if we are allowed to 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 the condition (7) or a null step with the condition (5) occurs (see [41]). Nevertheless, in our numerical experiments the condition (5) was always satisfied with the value $t = 1$ and no line search was needed.

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

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

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

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 [20, 21]. 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

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

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 $\mu_{max} > \mu_{min} > 0$, 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 $w_k = \tilde{\boldsymbol{\xi}}_k^T D_k^+ \tilde{\boldsymbol{\xi}}_k + 2\tilde{\beta}_k$. If $w_k < \varepsilon$, then stop with $\boldsymbol{x}_k$ as the final solution.

Step 4: (*Auxiliary Step*) Evaluate

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

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, add 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 $\beta_{k+1}$ as in (6). If

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

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 corresbonding 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 $\beta_{k+1}$ 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 4.1. The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is semismooth (see e.g. [8]), which ensures

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

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

ASSUMPTION 4.2. 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 4.1. 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 not sufficient without some convexity assumptions, and the objective 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 $\varepsilon$ is equal to zero. As before, we drop out the index $t$ from $\boldsymbol{y}_k^t$, $\boldsymbol{\xi}_k^t$ and $\beta_k^t$ even if $t \neq 1$.

REMARK 4.2. The sequence $(\boldsymbol{x}_k)$ generated by Algorithm 3.1 is bounded by Assumption 4.2 and taking into account the monotonicity of the sequence $(f_k)$ which, in turn, 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 closed interval $[\mu_{min}, \mu_{\max}]$ ($[-\mu_{\max}, -\mu_{min}]$). Thus, the search direction $\boldsymbol{d}_k$ 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 $k$th iteration of Algorithm 3.1, we have*

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

*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 the equations (6), (11), and Step 1 in Algorithm 3.1. The relations

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

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

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

□

LEMMA 4.2. *Suppose that Algorithm 3.1 is not terminated before the $k$th 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 [41]. □

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 [41]. □

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

THEOREM 4.4. *If Algorithm 3.1 terminates at the $k$th 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, $\tilde{\boldsymbol{\xi}}_k = \mathbf{0}$ 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 [41]. □

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*

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

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

*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, the condition (13) is valid.

Furthermore, we have

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

for all $k$, since $D_k^+$ is a diagonal matrix with the largest diagonal element less or equal to $\mu_{\max}$. Therefore, also the 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

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

for $k > m$.

Let us denote $D_k^+ = W_k^T W_k$. Then, the function $\varphi$ (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 4.2 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 similar to the proof (part $(ii)$) of Lemma 3.6 in [41]. □

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 $(f(\boldsymbol{x}_k))_{k\in\mathcal{K}'} \to f(\bar{\boldsymbol{x}})$ and, 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 the 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 $\mathcal{K}_1 = \{\, k \in \mathcal{K}' \mid t \geq t_{\min}\}$ is infinite for some bound $t_{\min} > 0$ then $(w_k)_{k\in\mathcal{K}'} \to 0$ and $(\boldsymbol{x}_k)_{k\in\mathcal{K}'} \to \bar{\boldsymbol{x}}$ by (16) and, thus, 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 [41]. □

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 extensions of classical academic nonsmooth minimization problems [20]. These problems can be formulated with any number of variables. We have tested these problems with 1000, 10000, 100000 and million variables. The other numerical experiments comparing different NSO solvers including the LMBM and the D-BUNDLE can be found for instance in [5, 27, 28].

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

`LMBM` is an implementation of the LMBM [20, 21] specifically developed for large-scale NSO. The solver uses the modified weak Wolfe -type line search for finding suitable step sizes (see, [21]). 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 [27]). Otherwise, the default parameters of the code were used.

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 a diagonal bundle solver developed specially for sparse nonsmooth minimization [27]. Similarly to `LMBM`, the solver uses the modified weak Wolfe -type line search to find suitable step sizes (see, [21]). 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

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

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 as a 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 the 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 take a null step. Note that no theoretical guarantee of the satiety of the null step condition (5) after these Armijo-type line searches is given but, in our numerical experiments it was always satisfied nonetheless.

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 search 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

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

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*. 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. Here, problems 1 – 5 are convex while problems 6 – 10 are nonconvex. 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. The asterix after a result means that the result obtained was inaccurate. Note that with the nonconvex problems it was not always easy to say, wether the solution obtained was a local solution (or a stationary point) or not. Thus, we have only accepted the results that convergent to the global minimum.

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 | 37 728/3.98* | 6 136/0.31 | 63 858/2.84 | **3 002/0.12** | 5 999/0.22 |
| 2 | fail | fail | fail | fail | fail |
| 3 | 3 292/0.13 | 3 751/0.10 | 61 436/3.60 | **222/0.01** | 918/0.04 |
| 4 | 3 450/0.16 | 6 917/0.25 | 81 449/5.44 | **240/0.01** | 627/0.03 |
| 5 | 326/0.02 | 1 388/0.04 | fail | **223/0.01** | 719/0.04 |
| 6 | 1 138/0.05 | 1 075/0.07 | 1 719/0.09 | **710/0.02** | 983/0.05 |
| 7 | 5 690/0.79 | 13 319/1.19 | **246/0.04** | 663/0.06 | 1225/0.18 |
| 8 | **6 020**/0.21 | 7 617/**0.17** | 1 000 166/50.96 | 1 999 991/63.41 | 192 513/3.52 |
| 9 | 1 128/0.05 | 1 106/0.04 | fail | **112/0.00** | fail |
| 10 | 11 282/0.39 | 17 377/0.49 | 454/0.03 | **338/0.02** | 1 302/0.07 |

With 1000 variables the new solver SMDBA with few rounds of Armijo line search was superior when compared with other methods (see Table 1). Nevertheless, the test set is not large enough to draw any far reaching superiority assumptions. While SMDBA seems 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.

With 10 000 variables the superiority of SMDBA is not at all clear anymore (see Table 2). In fact, here both SMDB and SMDBNM seem to be more efficient than SMDBA when only nonconvex problems are considered and the efficiency of SMDBA and SMDBNB were quite similar in convex problems.

A problem with 100 000 variables can be considered as an extremely large nonsmooth problem. Here, D-Bundle was clearly the most efficient solver when comparing convex problems (see Problems 1–5 in Table 3). However, as all the other solvers but SMDBNB , it failed to solve Problems 1 and 2 and thus the sample set is very small. With nonconvex problems (Problems 6–10 in Table 3) there was quite a big variation

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 | 179 502/100.78 | 913 677/392.26 | **30 014/10.79** | 60 003/21.81 |
| 2 | fail | fail | fail | fail | fail |
| 3 | 6 082/2.11 | 3 669/0.93 | 31 482/17.31 | **311/0.13** | 1 059/0.45 |
| 4 | 7 080/2.93 | **5 144/1.90** | 1 000 104/654.80 | 528 300/234.68 | 9 007/3.46 |
| 5 | **244**/0.17 | 307/**0.11** | fail | 5 643/2.22 | 792/0.39 |
| 6 | 10 108/4.25 | 10 104/6.32 | 15 197/7.55 | 6 421/2.39 | **1 175/0.47** |
| 7 | 8 888/11.27 | 49 823/41.69 | **360/0.61** | 1 186/1.53 | 895/1.28 |
| 8 | **6 232/2.40** | 11 261/2.64 | 1 000 182/500.13 | 1 999 992/627.62 | 361 347/60.66 |
| 9 | fail | 474/0.24* | fail | fail | fail |
| 10 | fail | fail | 3 232/1.76 | 113 090/41.65 | **1 197/0.58** |

in the performance of the solvers with different problems and we can not say that one solver was better than the other. More so, since similarly to convex problems the solvers succeed in solving at most three problems out of five with the desired accuracy. However, as said before, with the nonconvex problems it is not always easy to say, whether the solution obtained was 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 | **396 980/1 367.28** |
| 2 | fail | fail | fail | fail | fail |
| 3 | 5 796/18.30 | **144/0.35** | 1 000 106/5 165.92 | 5 145/17.22 | 1 270/5.05 |
| 4 | 10 424/39.79 | **584/2.43** | 1 000 062/6 010.81 | 1 609 217/7 200.00 | 5 125/19.27 |
| 5 | **438**/3.13 | 816/**2.47** | fail | 8 475/34.47 | 772/3.62 |
| 6 | 100 142/**386.55** | **100 100**/652.58 | fail | fail | fail |
| 7 | fail | 53 905/452.56 | **310/5.16** | 7 188/94.80 | 2 251/25.44 |
| 8 | **2 100/12.76** | 5 141/15.09 | 1 000 131/4 702.07 | 1 999 993/6 191.57 | 779 804/1 177.88 |
| 9 | **1 400/11.44** | 1 086/6.36* | fail | fail | fail |
| 10 | 34 630/116.87* | fail | 14 755/77.96 | 5 034/18.39 | **1 757/7.80** |

With million variables SMDBNM succeed in solving six problems out of ten being the most robust of the solvers. All the other solvers succeed in solving at most five problems out of ten, although, all the sovers converged to the same (stationary?) point also in Problem 6. One could say, that solving only five or six problem is not very convincing but it is better than most of nonsmooth algorithms can do [5, 27, 28].

The accuracies of the tested solvers were quite similar but SMDB was the least accurate solver especially with 1 000 variables (see Table 1)). This is probably due to the lack of any line search. Already two rounds of Armijo -type line search made a difference. Nevertheless, the differences in accuracy were not as clear with larger problems and indeed with million variables and nonconvex problems SMDB was in fact the most robust solver together with SMDBNM. In all given numbers of variables all the solvers failed to solve Problem 2. In addition, LMBM always failed to solve Problem

Table 4: Summary of the results with million variables.

| P | LMBM<br>$nfg/cpu$ | D-bundle<br>$nfg/cpu$ | SMDB<br>$nfg/cpu$ | SMDBA<br>$nfg/cpu$ | SMDBNM<br>$nfg/cpu$ |
|---|---|---|---|---|---|
| 1 | fail | fail | fail | fail | fail |
| 2 | fail | fail | fail | fail | fail |
| 3 | 1 698/59.76 | **367/12.03** | 150 697/7 200.07 | 232 510/7 200.05 | 103 873/1 730.92 |
| 4 | 14 302/552.34 | 28 099/953.67 | 119 671/7 200.10 | 170 700/7 200.07 | **3 698/141.81** |
| 5 | 1 580/105.69 | **772/24.32** | fail | 70 024/2 822.73 | 11 204/323.90 |
| 6 | fail | fail | fail | fail | fail |
| 7 | fail | fail | **317/53.04** | fail | 2 789/294.10 |
| 8 | **2 632/252.66** | 19 727/918.20 | 153 163/7 200.07 | 242 014/7 200.00 | 119 455/2 035.17 |
| 9 | **2748/285.72** | 3 569/400.99* | fail | fail | fail |
| 10 | fail | 14 521/932.18* | **41 301/2 170.33** | 198 275/7 200.03 | 345 119/7 200.27 |

1 with the desired accuracy. The other solvers did a little bit better and succeed in solving this problem with 1 000 and 10 000 variables and SMDBNS also with 100 000 variables. 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.

# 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 or 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 all these versions are efficient for both convex (problems 1 – 5) and nonconvex (problems 6 – 10) nonsmooth optimization problems. With large 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 numbers of variables it needed more function and subgradient evaluations than those other solvers. On the other hand, with 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 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

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 in 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] 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.

[17] 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.

[18] 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.

[19] GAUDIOSO, M., AND GORGONE, E. Gradient set splitting in nonconvex nonsmooth numerical optimization. *Optimization Methods and Software 25* (2010), 59–74.

[20] 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.

[21] 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.

[22] HARE, W., AND SAGASTIZÁBAL, C. A redistributed proximal bundle method for nonconvex optimization. *SIAM Journal on Optimization 20*, 5 (2010), 2442–2473.

[23] HASLINGER, J., AND NEITTAANMÄKI, P. *Finite Element Approximation for Optimal Shape, Material and Topology Design*, 2nd edition ed. John Wiley & Sons, Chichester, 1996.

[24] 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).

[25] HIRIART-URRUTY, J.-B., AND LEMARÉCHAL, C. *Convex Analysis and Minimization Algorithms II*. Springer-Verlag, Berlin, 1993.

[26] KÄRKKÄINEN, T., AND HEIKKOLA, E. Robust formulations for training multilayer perceptrons. *Neural Computation 16* (2004), 837–862.

[27] 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.

[28] KARMITSA, N., BAGIROV, A., AND MÄKELÄ, M. M. Comparing different nonsmooth optimization methods and software. *Optimization Methods and Software 27*, 1 (2012), 131–153.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] MÄKELÄ, M. M., AND NEITTAANMÄKI, P. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific Publishing Co., Singapore, 1992.

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

[37] 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.

[38] MOREAU, J., PANAGIOTOPOULOS, P. D., AND STRANG, G., Eds. *Topics in Nonsmooth Mechanics*. Birkhäuser Verlag, Basel, 1988.

[39] 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.

[40] 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.

[41] 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.

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

**University of Turku**
*Faculty of Mathematics and Natural Sciences*
- Department of Information Technology
- Department of Mathematics and Statistics

*Turku School of Economics*
- Institute of Information Systems Sciences

**Åbo Akademi University**
- Computer Science
- Computer Engineering

ISBN 978-952-12-3530-6
ISSN 1239-1891