# Limited Memory Discrete Gradient Bundle Method for Nonsmooth Derivative Free Optimization

N. Karmitsa[1] and A.M. Bagirov[2]

*Abstract:* Typically, practical nonsmooth optimization problems involve functions with hundreds of variables. Moreover, there are many practical problems where the computation of even one subgradient is either a difficult or an impossible task. In such cases derivative free methods are the better (or only) choice since they do not use explicit computation of subgradients. However, these methods require a large number of function evaluations even for moderately large problems. In this paper, we propose an efficient derivative free limited memory discrete gradient bundle method for nonsmooth, possibly nonconvex optimization. The convergence of the proposed method is proved for locally Lipschitz continuous functions and the numerical experiments to be presented confirm the usability of the method especially for medium size and large-scale problems.

*Keywords:* Nondifferentiable optimization, derivative free optimization, limited memory methods, bundle methods, discrete gradient.

## 1 Introduction

We introduce a new derivative free method for solving unconstrained minimization problems of the form

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

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is supposed to be locally Lipschitz continuous (LLC). In particular, our aim is to design a derivative free method for solving Problem (P) with the size from medium to large (i.e. problems with more than 50 variables). Note that no differentiability or convexity assumptions are made and we do not assume that a (sub)gradient of the objective function can be evaluated.

Nonsmooth optimization (NSO) problems (P) are encountered in many application areas: for instance, in economics [40], mechanics [39], engineering [38], control theory [12], optimal shape design [22], machine learning [25], and data mining [2, 8] including cluster analysis [14]. Most of these problems are large-scale.

NSO problems are in general difficult to solve. The direct application of gradient-based methods to nonsmooth problems may lead to a failure in convergence, in optimality conditions, or in gradient approximation (see, e.g. [30]).

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

2. Centre for Informatics and Applied Optimization, School of Science, Information Technology and Engineering, University of Ballarat, University Drive, Mount Helen, PO Box 663, Ballarat, VIC 3353, Australia. E-mail: a.bagirov@ballarat.edu.au

Methods for solving NSO problems include subgradient methods (see e.g. [5, 6, 44, 45]), bundle methods (see e.g. [16, 23, 28, 32, 34, 35, 42, 43]), algorithms based on smoothing techniques [41], and the gradient sampling methods [9]. In most of these algorithms the computation of at least one arbitrary subgradient (generalized gradient [11]) at each iteration is required. However, there are many practical problems where computation of even one subgradient is a difficult (or an impossible) task. In such cases derivative free methods are the preferable (or only) choice since they do not require explicit computation of subgradients.

Most of existing derivative free methods like the genetic algorithm (see, e.g. [17]) or Powell's method (see, e.g. [15]) are inefficient for solving nonsmooth optimization problems with even tens of variables. For the other derivative free methods the convergence can only be proved under differentiability or strict differentiability assumptions (see, e.g. [1, 13, 15]). The exceptions are the discrete gradient method [4] and the approximative subgradient method [3], where the convergence are proved for quasidifferentiable semismooth functions.

In this paper we introduce a new limited memory discrete gradient bundle method for derivative free nonsmooth optimization. The idea of the new method is to combine the discrete gradient method DGM [4] and the limited memory bundle method LMBM [19, 20, 21]. DGM is a derivative free method for (small-scale) nonconvex nonsmooth optimization while LMBM utilizes subgradient information to solve large-scale nonsmooth problems. Although these two are totally different methods they have some similarities in their structures that makes combining of them rather an interesting task. The convergence of the proposed method is proved for semismooth functions.

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

## 2 Background

In this section, we first recall some basic definitions and results from nonsmooth analysis. Then, we discuss basic ideas of LMBM and DGM.

### 2.1 Preliminaries

We denote by $\|\cdot\|$ the Euclidean norm in $\mathbb{R}^n$ and by $\boldsymbol{a}^T\boldsymbol{b}$ the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ (bolded symbols are used for vectors). An open (closed) ball with center $\boldsymbol{x} \in \mathbb{R}^n$ and radius $r > 0$ is denoted by $B(\boldsymbol{x}; r)$ ($\bar{B}(\boldsymbol{x}; r)$). That is,

$$B(\boldsymbol{x}; r) = \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{x}\| < r\} \quad \text{and} \quad \bar{B}(\boldsymbol{x}; r) = \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{x}\| \leq r\}.$$

The *subdifferential* $\partial f(\boldsymbol{x})$ [11] of a LLC function $f : \mathbb{R}^n \to \mathbb{R}$ at any point $\boldsymbol{x} \in \mathbb{R}^n$ is given by

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

where "conv" denotes the convex hull of a set. A vector $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is called a *subgradient*. The *Goldstein $\varepsilon$-subdifferential* [18]) with some $\varepsilon \geq 0$ is a set

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

Note that $\partial f(\boldsymbol{x}) \subseteq \partial_\varepsilon^G f(\boldsymbol{x})$ for all $\varepsilon \geq 0$.

The point $\boldsymbol{x}^* \in \mathbb{R}^n$ is called *substationary* if $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. Substationarity 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 substationary point $\boldsymbol{x}^*$, that is, $\{\boldsymbol{x}_k\} \to \boldsymbol{x}^*$ whenever $k \to \infty$.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *semismooth* at $\boldsymbol{x} \in \mathbb{R}^n$ if it is LLC at $\boldsymbol{x}$ and for every $\boldsymbol{d} \in \mathbb{R}^n$ the limit

$$\lim_{\substack{\xi \in \partial f(\boldsymbol{x}+h\boldsymbol{d}'),\\ \boldsymbol{d}' \to \boldsymbol{d},\, h \to 0^+}} \boldsymbol{\xi}^T \boldsymbol{d}$$

exists [36]. If the function $f$ is semismooth, the classical directional derivative $f'(\boldsymbol{x}, \boldsymbol{d}) = \lim_{h \to 0^+} h^{-1}[f(\boldsymbol{x} + h\boldsymbol{d}) - f(\boldsymbol{x})]$ exists and

$$f'(\boldsymbol{x}, \boldsymbol{d}) = \lim_{\substack{\xi \in \partial f(\boldsymbol{x}+h\boldsymbol{d}'),\\ \boldsymbol{d}' \to \boldsymbol{d},\, h \to 0^+}} \boldsymbol{\xi}^T \boldsymbol{d}.$$

The class of semismooth functions contains convex, concave, max-type and min-type functions inter alia.

Let us denote by

$$S_1 = \{\boldsymbol{g} \in \mathbb{R}^n \mid \|\boldsymbol{g}\| = 1\}$$

the sphere of the unit ball and by

$$P = \{z \mid z : \mathbb{R}_+ \to \mathbb{R}_+,\ \zeta > 0,\ \zeta^{-1} z(\zeta) \to 0,\ \zeta \to 0\}$$

the set of univariate positive infinitesimal functions. In addition, let

$$G = \{\boldsymbol{e} \in \mathbb{R}^n \mid \boldsymbol{e} = (e_1, \ldots, e_n), |e_j| = 1,\ j = 1, \ldots, n\}$$

be a set of all vertices of the unit hypercube in $\mathbb{R}^n$. We take any $\boldsymbol{g} \in S_1$, $\boldsymbol{e} \in G$, $z \in P$, a positive number $\alpha \in (0, 1]$, and compute $i = \operatorname{argmax}\{|g_j| \mid j = 1, \ldots, n\}$. For $\boldsymbol{e} \in G$ we define the sequence of $n$ vectors $\boldsymbol{e}^j(\alpha) = (\alpha e_1, \alpha^2 e_2, \ldots, \alpha^j e_j, 0, \ldots, 0)$ $j = 1, \ldots, n$. For $\boldsymbol{x} \in \mathbb{R}^n$ and $\zeta > 0$, consider the points

$$\boldsymbol{x}_0 = \boldsymbol{x} + \zeta \boldsymbol{g}, \qquad \boldsymbol{x}_j = \boldsymbol{x}_0 + z(\zeta)\boldsymbol{e}^j(\alpha), \qquad j = 1, \ldots, n. \tag{1}$$

Definition 2.1. Let $\boldsymbol{g} \in S_1$, $\boldsymbol{e} \in G$, $z \in P$, $\alpha \in (0, 1]$, $\zeta > 0$ and take $i = \operatorname{argmax}\{|g_j| \mid j = 1, \ldots, n\}$. The *discrete gradient* of the function $f : \mathbb{R}^n \to \mathbb{R}$ at the point $\boldsymbol{x} \in \mathbb{R}^n$ in the direction $\boldsymbol{g}$ is the vector $\Gamma^i(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{e}, z, \zeta, \alpha) = (\Gamma_1^i, \ldots, \Gamma_n^i) \in \mathbb{R}^n$ with the following coordinates:

$$\Gamma_j^i = [z(\zeta)\alpha^j e_j)]^{-1} \left[f(\boldsymbol{x}_j) - f(\boldsymbol{x}_{j-1})\right], \qquad j = 1, \ldots, n, \quad j \neq i,$$

$$\Gamma_i^i = (\zeta g_i)^{-1} \left[f(\boldsymbol{x} + \zeta \boldsymbol{g}) - f(\boldsymbol{x}) - \zeta \sum_{j=1, j \neq i}^{n} \Gamma_j^i g_j\right].$$

It follows from Definition 2.1 that

$$f(\boldsymbol{x}+\zeta\boldsymbol{g})-f(\boldsymbol{x})=\zeta\boldsymbol{g}^T\Gamma^i(\boldsymbol{x},\boldsymbol{g},\boldsymbol{e},z,\zeta,\alpha) \tag{2}$$

for all $\boldsymbol{g}\in S_1$, $\boldsymbol{e}\in G$, $z\in P$, $\alpha\in(0,1]$ and $\zeta>0$.

REMARK 2.1. The discrete gradient is defined with respect to a given direction $\boldsymbol{g}\in S_1$. To compute the discrete gradient we first define a sequence of points $\boldsymbol{x}_0,\dots\boldsymbol{x}_n$ (see (1)) and compute the values of the function $f$ at these point. That is, we compute $n+2$ values of $f$ including the point $\boldsymbol{x}$. The $i$th coordinate is defined so that it satisfies equality (2) which can be considered as some version of the mean value theorem.

It has been proved in [4] that for any $\boldsymbol{x}\in\mathbb{R}^n$, $\boldsymbol{g}\in S_1$, $\boldsymbol{e}\in G$, $\zeta>0$, $\boldsymbol{z}\in P$, and $\alpha>0$ we have $\|\Gamma^i\|\le C(n)L$, where $C(n)=(n^2+2n^{3/2}-2n^{1/2})^{1/2}$ and $L$ is a Lipschitz constant of function $f$ at $\boldsymbol{x}$. Furthermore, the closed convex set of discrete gradients

$$V_0(\boldsymbol{x},\zeta)=\operatorname{cl}\operatorname{conv}\{\boldsymbol{v}\in\mathbb{R}^n\mid\exists\ (\boldsymbol{g}\in S_1,\ \boldsymbol{e}\in G,\ z\in P,\alpha>0)\\ \text{such that } \boldsymbol{v}=\Gamma^i(\boldsymbol{x},\boldsymbol{g},\boldsymbol{e},z,\zeta,\alpha)\}$$

is an approximation to the subdifferential $\partial f(\boldsymbol{x})$ for sufficiently small $\zeta>0$ as stated in the following proposition [4]:

PROPOSITION 2.2. *Let $f:\mathbb{R}^n\to\mathbb{R}$ be a semismooth function at $\boldsymbol{x}$. For $\zeta>0$ and $\boldsymbol{g}\in S_1$ define*

$$o(\zeta,\boldsymbol{g})=f(\boldsymbol{x}+\zeta\boldsymbol{g})-f(\boldsymbol{x})-\zeta f'(\boldsymbol{x},\boldsymbol{g}).$$

*If $\zeta^{-1}o(\zeta,\boldsymbol{g})\to 0$ uniformly with respect to $\boldsymbol{g}$ as $\zeta\to 0^+$, then for any $\varepsilon>0$ there exists $\zeta_0>0$ such that*

$$V_0(\boldsymbol{x},\zeta)\subset\partial f(\boldsymbol{x})+B(\boldsymbol{0};\varepsilon)$$

*for all $\zeta\in(0,\zeta_0)$.*

### 2.2 Discrete Gradient Method

Next we briefly describe the discrete gradient method (DGM) by Bagirov et. al. More details can be found in [4]. The idea of DGM is to hybridize derivative free methods with bundle methods. In contrast with bundle methods, which require the computation of a single subgradient of the objective function at each trial point, DGM approximates subgradients by discrete gradients using function values only. Similarly to bundle methods the previous values of discrete gradients are gathered into a bundle and the null step is used if the current search direction is not good enough.

We start by describing the direction finding procedure. As mentioned before, the closed convex set of discrete gradients $V_0(\boldsymbol{x},\zeta)$ is an approximation to the subdifferential $\partial f(\boldsymbol{x})$ for sufficiently small $\zeta>0$. Thus, it can be used to compute the descent direction for the objective. However, the computation of the whole set $V_0(\boldsymbol{x},\zeta)$ is not easy, and therefore, in DGM only a few discrete gradients from the set are used to calculate the descent direction.

The DGM consists of an outer and an inner iterations. Let us denote by $k$ the index of the outer iteration and by $s$ the index of inner iteration. In what follows we use only the iteration counter $s$ whenever possible without confusion. At every iteration $k$ we first (i.e. $s = 1$) compute the discrete gradient $\boldsymbol{v}_1 = \Gamma^i(\boldsymbol{x}_{k_1}, \boldsymbol{g}_1, \boldsymbol{e}, z, \zeta, \alpha)$ (see Definition 2.1) with respect to any initial direction $\boldsymbol{g}_1 \in S_1$ and we set the initial bundle of discrete gradients $\bar{V}_1(\boldsymbol{x}_{k_1}) = \{\boldsymbol{v}_1\}$. Then we compute the vector

$$\bar{\boldsymbol{v}}_s = \text{argmin}_{\,\boldsymbol{v} \in \bar{V}_s(\boldsymbol{x}_{k_s})} \|\boldsymbol{v}\|^2,$$

that is the distance between the convex hull $\bar{V}_s(\boldsymbol{x}_{k_s})$ of all computed discrete gradients and the origin. If this distance is less than a given tolerance $\delta > 0$ we accept the point $\boldsymbol{x}_{k_s}$ as an approximate substationary point and go to the next outer iteration. Otherwise, we compute another search direction

$$\boldsymbol{g}_{s+1} = -\frac{\bar{\boldsymbol{v}}_s}{\|\bar{\boldsymbol{v}}_s\|}$$

and we check whether this is a descent direction. That is, we have

$$f(\boldsymbol{x} + \zeta \boldsymbol{g}_{s+1}) - f(\boldsymbol{x}) \leq -c_1 \zeta \|\bar{\boldsymbol{v}}_s\|,$$

with the given numbers $c_1 \in (0, 1)$ and $\zeta > 0$. In this case, we set $\boldsymbol{d}_{k_s} = \boldsymbol{g}_{s+1}$ and stop the direction finding procedure. Otherwise, we take a null step. That is, we compute another discrete gradient $\boldsymbol{v}_{s+1} = \Gamma^i(\boldsymbol{x}_{k_s}, \boldsymbol{g}_{s+1}, \boldsymbol{e}, z, \zeta, \alpha)$ into the direction $\boldsymbol{g}_{s+1}$, update the bundle of discrete gradients by

$$\bar{V}_{s+1}(\boldsymbol{x}_{k_s}) = \text{conv}\{\bar{V}_s(\boldsymbol{x}_{k_s}) \cup \{\boldsymbol{v}_{s+1}\}\}$$

and continue the direction finding procedure with $\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s}$ and $s = s+1$. Note that, at the null steps the approximation of the subdifferential $\partial f(\boldsymbol{x})$ is improved. It has been proved in [4] that the direction finding procedure is terminating such that either the descent direction $\boldsymbol{d}_{k_s}$ is found or $\|\bar{\boldsymbol{v}}_s\| \leq \delta$.

When the descent direction $\boldsymbol{d}_{k_s}$ has been found, we need to compute the next (inner) iteration point $\boldsymbol{x}_{k_{s+1}} = \boldsymbol{x}_{k_s} + t_{k_s} \boldsymbol{d}_{k_s}$, where the step size $t_{k_s}$ is defined as

$$t_{k_s} = \text{argmax}\, \{t \geq 0 \mid f(\boldsymbol{x}_{k_s} + t\boldsymbol{d}_{k_s}) - f(\boldsymbol{x}_{k_s}) \leq -c_2 t \|\bar{\boldsymbol{v}}_{k_s}\|\}$$

with given $c_2 \in (0, c_1]$.

The pseudo-code of DGM is the following:

```
Program DGM
  Initialize x_1 ∈ R^n and k = 1;
  Outer iteration
    Set s = 1 and x_{k_s} = x_k;
    Compute the first discrete gradient v_{k_1};
    Set V̄(x_{k_1}) = {v_{k_1}};
    While the termination condition is not met
      Inner iteration
        Compute the vector v̄_{k_s} = argmin_{v ∈ V̄(x_{k_s})} ‖v‖^2;
        If ‖v̄_{k_s}‖ ≤ δ_k with δ_k > 0 s.t. δ_k ↘ 0 when k → ∞ then
          Set x_{k+1} = x_{k_s} and k = k + 1;
          Go to the next Outer iteration;
        Else
          Compute the search direction d_{k_s} = −v̄_{k_s}/‖v̄_{k_s}‖;
          Find a step size t^k;
          If Descent condition holds then
            Construct the following iteration x_{k_{s+1}} = x_{k_s} + t^k d_{k_s};
            Compute a new discrete gradient v_{k_{s+1}} at x_{k_{s+1}};
            Set V̄(x_{k_{s+1}}) = {v_{k_{s+1}}};
          Else
            Compute a new discrete gradient v_{k_{s+1}} at x_{k_s}
              in direction d_{k_s};
            Update the set V̄(x_{k_{s+1}}) = conv{ V̄(x_{k_s}) ∪ v_{k_{s+1}} };
            Set x_{k_{s+1}} = x_{k_s};
          End If
          Set s = s + 1 and go to the next Inner iteration;
        End If
      End Inner iteration
    End While
  End Outer iteration
  Return final solution x_k;
End DGM
```

It has been proved that DGM is globally convergent for quasidifferentiable LLC functions under the assumption that the set of discrete gradients uniformly approximates the subdifferential [4].

### 2.3 Limited Memory Bundle Method (LMBM)

In this subsection, we describe the limited memory bundle algorithm (LMBM) by Karmitsa (née Haarala) et. al. [19, 20, 21, 27] for solving general, possibly nonconvex, large-scale NSO problems. Here we assume that at every point $\boldsymbol{x}$ we can evaluate the value of the objective function $f(\boldsymbol{x})$ and one arbitrary subgradient $\boldsymbol{\xi}$ from the subdifferential $\partial f(\boldsymbol{x})$.

LMBM is a hybrid of the variable metric bundle methods [32, 46] and the limited memory variable metric methods (see e.g. [10]), where the first ones have been developed for small- and medium-scale nonsmooth optimization and the latter ones for smooth large-scale optimization. LMBM exploits the ideas of the variable metric bundle methods, namely the utilization of null steps, simple aggregation of subgradients, and the subgradient locality measures, but the search

direction $\boldsymbol{d}_k$ is calculated using the limited memory approach. That is,

$$\boldsymbol{d}_k = -D_k \tilde{\boldsymbol{\xi}}_k,$$

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

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

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k \qquad \text{and} \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k, \qquad \text{for } k \geq 1 \end{aligned} \tag{3}$$

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

$$t_R^k = t_L^k > 0 \qquad \text{and} \qquad f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L^k t_R^k w_k, \tag{4}$$

where $\varepsilon_L^k \in (0, 1/2)$ is a line search parameter and $w_k > 0$ represents the desirable amount of descent of $f$ at $\boldsymbol{x}_k$. If the condition (4) is satisfied, we have $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$. On the other hand, a *null step* is taken if

$$t_R^k > t_L^k = 0 \qquad \text{and} \qquad -\beta_{k+1} + \boldsymbol{d}_k^T \boldsymbol{\xi}_{k+1} \geq -\varepsilon_R^k w_k, \tag{5}$$

where $\varepsilon_R^k \in (\varepsilon_L^k, 1/2)$ is a line search parameter, $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, and $\beta_{k+1}$ is the subgradient locality measure [31, 37] similar to standard bundle methods. That is,

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

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

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

LMBM uses the original subgradient $\boldsymbol{\xi}_k$ after the serious step and the aggregate subgradient $\tilde{\boldsymbol{\xi}}_k$ after the null step for direction finding (i.e. we set $\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k$ if the previous step was a serious step). The aggregation procedure is carried out by determining multipliers $\lambda_i^k$ satisfying $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function

$$\begin{aligned} \varphi(\lambda_1, \lambda_2, \lambda_3) = [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k]^T D_k [\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k] \\ + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k). \end{aligned} \tag{7}$$

Here $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$ is the current subgradient ($m$ denotes the index of the iteration after the latest serious step, i.e. $\boldsymbol{x}_k = \boldsymbol{x}_m$), $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ is the auxiliary subgradient, and $\tilde{\boldsymbol{\xi}}_k$ is the current aggregate subgradient from the previous iteration ($\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$). In addition, $\beta_{k+1}$ is the current subgradient locality measure and

$\tilde{\beta}_k$ is the current aggregate subgradient locality measure ($\tilde{\beta}_1 = 0$). The resulting aggregate subgradient $\tilde{\boldsymbol{\xi}}_{k+1}$ and aggregate subgradient locality measure $\tilde{\beta}_{k+1}$ are computed by the formulae

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \qquad \text{and} \qquad \tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{8}$$

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

In LMBM both the limited memory BFGS (L-BFGS) and the limited memory SR1 (L-SR1) update formulae [10] are used in calculations of the search direction and the aggregate values. The idea of limited memory matrix updating is that instead of storing large $n \times n$ matrices $D_k$, one stores a certain (usually small) number of vectors obtained at the previous iterations of the algorithm, and uses these vectors to implicitly define the variable metric matrices. In the case of a null step, we use the L-SR1 update, since this update formula allows us to preserve the boundedness and some other properties of generated matrices which guarantee the global convergence of the method. Otherwise, since these properties are not required after a serious step, the more efficient L-BFGS update is employed (see, for more details, [19, 20, 21]).

As a stopping parameter, we use the value

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

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

The pseudo-code of LMBM is the following:

```
Program LMBM
  Initialize x_1 ∈ R^n, ξ_1 ∈ ∂f(x_1), and ε > 0;
  Set k = 1 and d_1 = −ξ_1;
  While the termination condition w_k ≤ ε is not met
    Find step sizes t_L^k and t_R^k;
    Set x_{k+1} = x_k + t_L^k d_k;
    Evaluate f(x_{k+1}) and ξ_{k+1} ∈ ∂f(x_k + t_R^k d_k);
    If t_L^k > 0 then
      Serious step
        Compute the search direction d_{k+1} using ξ_{k+1} and L-BFGS
          update;
      End Serious step
    Else
      Null step
        Compute the aggregate subgradient ξ̃_{k+1};
        Compute the search direction d_{k+1} using ξ̃_{k+1} and L-SR1
          update;
      End Null step
    End If
    Set k = k + 1;
  End While
  Return final solution x_k;
End LMBM
```

Under the upper semismoothness assumption [7] LMBM can be proved to be globally convergent for LLC objective functions [19, 21].

## 3 Method

In this section we introduce the new derivative free limited memory discrete gradient bundle method (LDGBM).

As mentioned in the introduction LMBM and DGM have some similarities in their structures. For instance, both of these methods wipe out the old information whenever the serious step occurs. This property is different from standard bundle methods (see e.g. [35]) where the old information is collected near the current iteration point and stored to be used in the next iterations nonetheless of the step in question. In practice, storing all the old information may have several disadvantages: first, it needs storage space; second, it adds computational costs; and, what is the worst, it may store and use information that is no longer relevant due to the fact that it might have been collected far away from the current iteration point. The last point may be especially problematic in nonconvex cases.

In LMBM we bundle the subgradients that are computed in a small neighborhood of the current iteration point. This is similar to standard bundle methods although in LMBM we use this information only after null steps and, at the most, three subgradients are needed. On the other hand, in DGM discrete gradients calculated only at the current iteration point with respect to different directions are gathered into a bundle (see Definition 2.1 and Subsection 2.2). In our new method we combine these ideas and compute discrete gradients in a small neighborhood of the current iteration point with respect to different directions.

In DGM , similar to standard bundle methods, a quadratic subproblem needs to be solved to find the discrete gradient with the shortest norm and, as a consequence, to calculate the search direction. Now, instead of bundling an unlimited number of discrete gradients in null steps and computing the shortest norm, we compute the convex combination of at most three discrete gradients and the search direction is calculated using limited memory approach. Thus, a possibly time consuming quadratic direction finding problem needs not to be solved and also the difficulty with the unbounded amount of storage needed in DGM has been dealt with.

The obvious difference between LDGBM and LMBM is that we now use discrete gradients instead of subgradients of the objective function. In addition, we use both inner and outer iterations in order to avoid too tight approximations to the subgradients at the beginning of computation (thus, we have a derivative free method). The inner iteration of LDGBM is essentially same as LMBM. That is, the search direction is computed by the formula

$$\boldsymbol{d}_{k_s} = -D_{k_s}\tilde{\boldsymbol{v}}_{k_s},$$

where $s$ and $k$ are the indices of inner and outer iterations, $\tilde{\boldsymbol{v}}_{k_s}$ is an aggregate discrete gradient and $D_{k_s}$ is a limited memory variable metric update. In addition, the line search procedure (cf. (3) – (5)) is used to determine a new iteration and auxiliary points $\boldsymbol{x}_{k_{s+1}}$ and $\boldsymbol{y}_{k_{s+1}}$, and the aggregation procedure (cf. (7) and (8)) is used to compute a new aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_{s+1}}$ and a new aggregate subgradient locality measure $\tilde{\beta}_{k_{s+1}}$.

The first discrete gradient $\boldsymbol{v}_{1_1} = \Gamma^i(\boldsymbol{x}, \boldsymbol{g}_{1_1}, \boldsymbol{e}, z, \zeta, \alpha)$, where $i = \operatorname{argmax}\{|g_j| \mid j = 1, \ldots, n\}$, is computed to an arbitrary initial direction $\boldsymbol{g}_{1_1} \in S_1$. After that we always use the previous normalized search direction $\boldsymbol{g}_{k_{s+1}} = \boldsymbol{d}_{k_s}/\|\boldsymbol{d}_{k_s}\|$ to

compute the next discrete gradient $\boldsymbol{v}_{k_{s+1}}$. Parameters $z \in P$, $\zeta > 0$, and $\alpha > 0$ are selected similarly to DGM [4].

The inner iteration is terminated if we have

$$\frac{1}{2}\|\tilde{\boldsymbol{v}}_{k_s}\|^2 + \tilde{\beta}_{k_s} \leq \delta_k$$

for some outer iteration parameter $\delta_k > 0$.

We use here a new adaptive updating strategy for the selection of outer iteration parameter $\delta_k$. At the beginning, the outer iteration parameter $\delta_1$ is set to a large number. Each time the inner iteration is terminated we set

$$\delta_{k+1} = \min\{\sigma\delta_k, w_{k_s}\},$$

where $\sigma \in (0,1)$ and $w_{k_s} = -\tilde{\boldsymbol{v}}_{k_s}^T \boldsymbol{d}_{k_s} + 2\tilde{\beta}_{k_s}$. Similarly to LMBM, the parameter $w_{k_s}$ is used also during the line search procedure to represent the desirable amount of descent (cf. (4) and (5)).

Let us assume that the sequences $z_k \in P$, $\zeta_k > 0$, $z_k \to 0^+$, $\zeta_k \to 0^+$, $k \to \infty$, a sufficiently small number $\alpha > 0$ and the line search parameters $\varepsilon_L^{k_s} \in (0, 1/2)$ and $\varepsilon_R^{k_s} \in (\varepsilon_L^{k_s}, 1/2)$ are given. The pseudo-code of LDGBM is the following (note that obviously more details are given here than before):

```
Program LDGBM
  Initialize x_1 ∈ R^n, g_{1_1} ∈ S_1, ε > 0, δ_1 > ε, ζ_1 > 0, σ, ϵ ∈ (0,1) and k = 1;
  Outer iteration
    Set s = 1 and x_{k_1} = x_k;
    While the termination condition δ_k ≤ ε is not met
      Inner iteration
        Serious step 1
          Compute the discrete gradient v_{k_s} ∈ V_0(x_{k_s}, ζ_k) in
            direction g_{k_s};
          Set m = s, ṽ_{k_s} = v_{k_s}, and β̃_{k_s} = 0;
          Compute the search direction d_{k_s} using ṽ_{k_s} and L-BFGS
            update;
        End Serious step 1
        Inner iteration termination
          If 1/2‖ṽ_{k_s}‖^2 + β̃_{k_s} ≤ δ_k then;
            Set x_{k+1} = x_{k_s}, g_{k+1_1} = d_{k_s}/‖d_{k_s}‖, ζ_{k+1} = ϵζ_k,
              δ_{k+1} = min{σδ_k, −ṽ_{k_s}^T d_{k_s} + 2β̃_{k_s}}, and k = k + 1;
            Go to the next Outer iteration;
          End If
        End Inner iteration termination
        Find step sizes t_L^{k_s} and t_R^{k_s}, and the subgradient
          locality measure β_{k_{s+1}};
        If t_L^{k_s} > 0 then
          Serious step 2
            Construct the iteration x_{k_{s+1}} = x_{k_s} + t_L^{k_s} d_{k_s};
            Set g_{k_{s+1}} = d_{k_s}/‖d_{k_s}‖;
            Set s = s + 1 and go to the next Serious step 1;
          End Serious step 2
        Else
          Null step
            Construct the trial point y_{k_{s+1}} = x_{k_s} + t_R^{k_s} d_{k_s};
            Compute the new discrete gradient v_{k_{s+1}} ∈ V_0(y_{k_{s+1}}, ζ_k)
              at the point y_{k_{s+1}} in the direction g_{k_{s+1}} = d_{k_s}/‖d_{k_s}‖;
            Compute the aggregate values
              ṽ_{k_{s+1}} = λ_1^{k_s} v_{k_m} + λ_2^{k_s} v_{k_{s+1}} + λ_3^{k_s} ṽ_{k_s}   and
              β̃_{k_{s+1}} = λ_2^{k_s} β_{k_{s+1}} + λ_3^{k_s} β̃_{k_s};
            Compute the new search direction d_{k_{s+1}} using ṽ_{k_{s+1}}
              and L-SR1 update;
            Set x_{k_{s+1}} = x_{k_s} and s = s + 1;
            Go to the Inner iteration termination;
          End Null step
        End if
      End Inner iteration
    End While
  End Outer iteration
  Return final solution x_k;
End LDGBM
```

The discrete gradient is computed according to Definition 2.1 and Remark 2.1. Similarly to LMBM the search direction and the aggregate values are computed by using the L-BFGS update after serious steps and L-SR1 update otherwise (see

[10] for more details of limited memory matrix updating and [20, 21] for their usage in LMBM). The step sizes $t_R^{k_s} \in (0, t_{max}]$ and $t_L^{k_s} \in [0, t_R^{k_s}$ with $t_{max} > 1$ are computed such that either condition (4) for serious steps or condition (5) for null steps is satisfied. In addition, the subgradient locality measure $\beta_{k_{s+1}}$ as well as the multipliers $\lambda_i^{k_s}$ satisfying $\lambda_i^{k_s} \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^{3} \lambda_i^{k_s} = 1$ utilized in the aggregation procedure are computed similarly to LMBM (see equations (6) and (7)).

### 3.1 Convergence Proof

We now prove the global convergence of the LDGBM-algorithm. In addition to assuming that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is LLC, the level set $\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1) \}$ is supposed to be bounded for every starting point $\boldsymbol{x}_1 \in \mathbb{R}^n$. Furthermore, we assume that each execution of the line search procedure is finite and the assumptions of Proposition 2.2 are satisfied. The line search procedure has been proved to be finite under the assumption of semismoothness (see [46]). On the other hand, in Proposition 2.2 we assume that function $f$ is semismooth. Since the class of semismooth functions includes the class of upper semismooth functions, we here assume that the objective function $f$ is semismooth.

Definition 3.1. Let $\varepsilon > 0$ and $\zeta > 0$. We define *the $\varepsilon$-set of discrete gradients* by

$$V_\varepsilon(\boldsymbol{x}, \zeta) = \operatorname{conv}\{V_0(\boldsymbol{y}, \zeta) \mid \boldsymbol{y} \in \bar{B}(\boldsymbol{x}; \varepsilon)\}.$$

Lemma 3.2. *For all inner iterations $k_s$, there exists $\varepsilon_{ks} \geq 0$ such that*

$$\tilde{\boldsymbol{v}}_{k_s} \in V_{\varepsilon_{ks}}(\boldsymbol{x}_{k_s}, \zeta_k).$$

Proof. After a serious step we have $\tilde{\boldsymbol{v}}_{k_{s+1}} = \boldsymbol{v}_{k_{s+1}} \in V_0(\boldsymbol{x}_{k_{s+1}}, \zeta_k)$ and, hence, the case is trivial. After a null step the aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_{s+1}}$ is computed as a convex combination of three discrete gradient: $\boldsymbol{v}_{k_s} \in V_0(\boldsymbol{x}_{k_s}, \zeta_k)$, $\boldsymbol{v}_{k_{s+1}} \in V_0(\boldsymbol{y}_{k_{s+1}}, \zeta_k)$ and $\tilde{\boldsymbol{v}}_{k_s}$. For null steps we use the scaled direction vector $\theta_k \boldsymbol{d}_k$ with $\theta_k = \min\{1, C/\|\boldsymbol{d}_k\|\}$ and predefined $C > 0$ in the line search (for more details, see [21]). Thus, by (3) we have $\|\boldsymbol{y}_{k_{s+1}} - \boldsymbol{x}_{k_s}\| \leq t_{max} C$, and we have $\boldsymbol{v}_{k_{s+1}} \in V_{\varepsilon_{ks}}(\boldsymbol{x}_{k_s}, \zeta_k)$ with some $0 < \varepsilon_{ks} \leq t_{max} C$. Since the first aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_s}$ after a serious step is equal to $\boldsymbol{v}_{k_s}$, they both belong to the set $V_{\varepsilon_{ks}}(\boldsymbol{x}_{k_s}, \zeta_k)$ with any $\varepsilon_{ks} \geq 0$.

As a convex combination of these three discrete gradients, the second aggregate discrete gradient $\tilde{\boldsymbol{v}}_{k_{s+1}}$ after the serious step belongs to the set $V_{\varepsilon_{ks}}(\boldsymbol{x}_{k_s}, \zeta_k)$ with some $0 < \varepsilon_{ks} \leq t_{max} C$. The rest of the proof follows by induction. □

Lemma 3.3. *Suppose that the assumptions of Proposition 2.2 are satisfied at every $\boldsymbol{y} \in \bar{B}(\boldsymbol{x}; \varepsilon)$ with some $\varepsilon > 0$. Then, there exists $\zeta_0 > 0$ such that the $\varepsilon$-set of discrete gradients approximates the Goldstein subdifferential by*

$$V_\varepsilon(\boldsymbol{x}, \zeta) \subset \partial_\varepsilon^G f(\boldsymbol{x}) + B(\boldsymbol{0}; \varepsilon)$$

*with all $\zeta \in (0, \zeta_0)$.*

Proof. Let $\boldsymbol{v} \in V_\varepsilon(\boldsymbol{x}, \zeta)$. Then $\boldsymbol{v} = \sum \lambda_i \boldsymbol{v}_i^{\boldsymbol{y}}$, where $\boldsymbol{v}_i^{\boldsymbol{y}} \in V_0(\boldsymbol{y}^i, \zeta)$, $\boldsymbol{y}^i \in \bar{B}(\boldsymbol{x}; \varepsilon)$, $\sum \lambda_i = 1$ and $\lambda_i \geq 0$ for all $i$. Now, by Proposition 2.2 there exists $\zeta_0^i$ for any

$\varepsilon > 0$ such that $V_0(\boldsymbol{y}^i, \zeta) \subset \partial f(\boldsymbol{y}^i) + B(\mathbf{0}; \varepsilon)$ for all $\zeta \in (0, \zeta_0^i)$. Thus, we have $\boldsymbol{v}_i^{\boldsymbol{y}} = \boldsymbol{\xi}_i^{\boldsymbol{y}} + \boldsymbol{s}_i^{\varepsilon}$, where $\boldsymbol{\xi}_i^{\boldsymbol{y}} \in \partial f(\boldsymbol{y}^i)$ and $\boldsymbol{s}_i^{\varepsilon} \in B(\mathbf{0}; \varepsilon)$. Hence,

$$\begin{aligned}
\boldsymbol{v} &= \sum \lambda_i (\boldsymbol{\xi}_i^{\boldsymbol{y}} + \boldsymbol{s}_i^{\varepsilon}) \\
&= \sum \lambda_i \boldsymbol{\xi}_i^{\boldsymbol{y}} + \sum \lambda_i \boldsymbol{s}_i^{\varepsilon} \\
&\subset \operatorname{conv}\{\partial f(\boldsymbol{y}^i) \mid \boldsymbol{y}^i \in \bar{B}(\boldsymbol{x}; \varepsilon)\} + B(\mathbf{0}; \varepsilon) \\
&= \partial_{\varepsilon}^{G} f(\boldsymbol{x}) + B(\mathbf{0}; \varepsilon)
\end{aligned}$$

with all $\zeta \in (0, \min_i\{\zeta_0^i\})$. □

Definition 3.4. A point $\boldsymbol{x}$ is called a $(\zeta, \delta)$-stationary point of the function $f$, if

$$\min_{\boldsymbol{v} \in V_0(\boldsymbol{x}, \zeta)} \|\boldsymbol{v}\| \leq \delta.$$

Theorem 3.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a LLC semismooth function. Then the number of inner iteration loops in the* LDGBM*-algorithm is finite and, at the termination, the point $x_{k+1}$ is a $(\zeta_k, \sqrt{2\delta_k})$ -stationary point of the function $f$.*

Proof. The inner iteration loop in the LDGBM-algorithm is essentially the same as LMBM with subgradients replaced by discrete gradients. It has been shown in [21] that LMBM terminates after finite number of iterations, if the stopping parameter (cf. $\delta_k$ here) is greater than zero. The usage of discrete gradients does not alter this result. Thus, the inner iteration loop in the LDGBM -algorithm is terminating after finite number of steps.

At the termination we have

$$\delta_k \geq \frac{1}{2}\|\tilde{\boldsymbol{v}}_{k_s}\|^2 + \tilde{\beta}_{k_s} \geq \frac{1}{2}\|\tilde{\boldsymbol{v}}_{k_s}\|^2.$$

By Lemma 3.2 we have $\tilde{\boldsymbol{v}}_{k_s} \in V_{\mu}(\boldsymbol{x}_{k_s}, \zeta_k)$ with some $\mu \geq 0$ and we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k_s}$ at the termination. Thus, we have

$$\min_{\boldsymbol{v} \in V_0(\boldsymbol{x}_{k+1}, \zeta_k)} \|\boldsymbol{v}\|^2 \leq \|\tilde{\boldsymbol{v}}_{k_s}\|^2 \leq 2\delta_k.$$

Then by Definition 3.4, $x_{k+1}$ is a $(\zeta_k, \sqrt{2\delta_k})$ -stationary point of the function $f$.□

Theorem 3.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a semismooth LLC function and suppose that the level set $\{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_1)\}$ is bounded. Then every accumulation point of the sequence $\{\boldsymbol{x}_k\}$ generated by the* LDGBM*-algorithm is substationary for $f$.*

Proof. Since $f$ is LLC and the level set is bounded, we have

$$f^* = \inf\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathbb{R}^n\} > -\infty.$$

By Theorem 3.5 the inner iterations generate $(\zeta_k, \sqrt{2\delta_k})$ -stationary points for all $k \geq 1$. Thus, it follows from Definition 3.4 that

$$\min\{\|\boldsymbol{v}\| \mid \boldsymbol{v} \in V_0(\boldsymbol{x}_{k+1}, \zeta_k)\} \leq \sqrt{2\delta_k} \tag{9}$$

for any $k \geq 1$. Since $\{f(\boldsymbol{x}_k)\}$ is a nonincreasing sequence, $\boldsymbol{x}_k$ belongs to the level set with all $k \geq 1$. Moreover, the boundedness of the level set implies that the

sequence $\{\boldsymbol{x}_k\}$ has at least one accumulation point. Let $\bar{\boldsymbol{x}}$ be an accumulation point of $\{\boldsymbol{x}_k\}$ and let $\boldsymbol{x}_{k_i} \to \bar{\boldsymbol{x}}$ when $i \to \infty$. Then we have from (9) that

$$\min\{\|\boldsymbol{v}\| \mid \boldsymbol{v} \in V_0(\boldsymbol{x}_{k_i}, \zeta_{k_i-1})\} \leq \sqrt{2\delta_{k_i-1}}. \tag{10}$$

Now, by Lemma 3.3 for any $\boldsymbol{y} \in \bar{B}(\bar{\boldsymbol{x}}; \mu)$ with $\mu > 0$ there exists $\zeta_0 > 0$ such that

$$V_0(\boldsymbol{y}, \zeta) \subseteq V_\mu(\bar{\boldsymbol{x}}, \zeta) \subseteq \partial_\mu^G f(\bar{\boldsymbol{x}}) + B(\boldsymbol{0}; \mu) \tag{11}$$

for all $\zeta \in (0, \zeta_0)$. Since $\boldsymbol{x}_{k_i}$ converges to $\bar{\boldsymbol{x}}$ there exists $i_0 \geq 1$ such that $\boldsymbol{x}_{k_i} \in B(\bar{\boldsymbol{x}}; \mu)$ for all $i \geq i_0$. Moreover, since $\delta_k \to 0$ and $\zeta_k \to 0$ as $k \to \infty$, there exists $k_0 > 1$ such that $\delta_k \leq \varepsilon$ with some $\varepsilon > 0$ and $\zeta_k < \zeta_0$ for all $k > k_0$. Then there exists $l_0 \geq i_0$ such that $k_l \geq k_0 + 1$ for all $l \geq l_0$. Therefore, from (10) and (11) we obtain

$$\min\{\|\boldsymbol{v}\| \mid \boldsymbol{v} \in \partial_\mu^G f(\bar{\boldsymbol{x}})\} \leq \mu + \sqrt{2\varepsilon}.$$

Now, since $\mu, \varepsilon > 0$ are arbitrary and the subdifferential is upper semicontinuous (see, e.g. [35]), we have $\boldsymbol{0} \in \partial f(\boldsymbol{x})$. □

## 4 Numerical Experiments

In this section we compare the implementations of the methods described in the previous sections and also SolvOpt, an implementation of Shor's $r$-algorithm. The test set used in our experiments consists of classical academic nonsmooth minimization problems from the literature and their extensions.

### 4.1 Solvers

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

Table 1: Tested pieces of software

| Software | Author(s) | Method | Reference |
|---|---|---|---|
| `SolvOpt` | Kuntsevich & Kappel | Shor's $r$-algorithm | [24, 29, 44] |
| `LMBM` | Karmitsa | Limited memory bundle | [20, 21] |
| `DGM` | Bagirov | Discrete gradient | [4] |
| `LDGBM` | Karmitsa | L-discrete gradient bundle | |

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

`SolvOpt` (Solver for local nonlinear optimization problems) is an implementation of Shor's $r$-algorithm. In `SolvOpt` one can select to use either original subgradients or difference approximations of them (i.e. the user does not have to code difference approximations but to select one parameter to do this automatically). In our experiments we have used difference approximations.

The MatLab, C and Fortran source codes for `SolvOpt` are available for downloading from http://www.kfunigraz.ac.at/imawww/kuntsevich/solvopt/. In our experiments we used `SolvOpt` v.1.1 HP-UX FORTRAN-90 sources.

`LMBM` is an implementation of the limited memory bundle method specifically developed for large-scale nonsmooth problems (see Subection 2.3). In our experiments we used the difference approximations to calculate the approximations of the subgradients. The Fortran 77 source code and the mex-driver (for MatLab users) are available for downloading from http://napsu.karmitsa.fi/lmbm/.

`DGM` is a discrete gradient solver (see Subection 2.2) for derivative free optimization. To apply `DGM`, one only needs to be able to compute at every point $\boldsymbol{x}$ the value of the objective function. The subgradient will then be approximated by discrete gradients. The Fortran 77 source code of `DGM` is available for downloading from http://napsu.karmitsa.fi/dgm/.

`LDGBM` is an implementation of the limited memory discrete gradient bundle method introduced in this paper. The Fortran 95 source code of `LDGBM` is available for downloading from http://napsu.karmitsa.fi/ldgbm/.

The experiments were performed on an Intel® Core™ 2 CPU 1.80GHz. To compile the codes, we used `gfortran-4.3`, the GNU Fortran compiler.

### 4.2 Test problems and parameters

As already said the test set used in our experiments consists of classical academic nonsmooth minimization problems from the literature and their extensions. That is, we have used the small-scale problems 3.1 – 3.20 of [33] (omitting problems 3.13 and 3.14 in which `DGM` converged to unbounded minima) and larger problems 1 – 10 introduced in [20] that can be formulated with any number of variables. We have used here 50, 200 and 1000 variables.

Note that in the computation of both the difference approximations and the discrete gradients more than $n$ function evaluations are needed per iteration. Therefore, already a problem with 200 variables can be considered as large one for the solvers not using (sub)gradient information and that is outstandingly true when dealing with nonsmooth problems.

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 small-scale problems ($n \leq 50$) with respect to the tolerance $\varepsilon = 5 \cdot 10^{-4}$. With larger problems ($n \geq 200$), we have accepted the results with the tolerance $\varepsilon = 10^{-3}$. To obtain comparable results the stopping parameters of the codes were tuned by the procedure similar to [26].

For `DGM` the maximum size of the bundle was set to $\min\{n+3, 100\}$. With `LMBM` and `LDGBM` the natural choice for the bundle size is two (the larger bundle may be used but it is only utilized in step size selection). For all other parameters we have used the default settings of the codes.

### 4.3 Results

The results are summarized in Tables 2 and 3. We have compared the efficiency of the solvers both in terms of the computational time (*cpu*) and the number of function evaluations ($nf$, evaluations for short). In the smaller problems ($n \leq 20$, see Table 2) we only give numbers of evaluations since there was not a big difference in the computational time of the different solvers.

Table 2: Summary of the results for small-scale problems

| P | $n$ | SolvOpt | LMBM | DGM | LDGBM |
|---|---|---|---|---|---|
| 3.1 | 2 | 270 | 989 | 1332 | 267 |
| 3.2 | 2 | 376 | 4 370 | 451 | 244 |
| 3.3 | 2 | 106 | 172 | 411 | 296 |
| 3.4 | 2 | 93 | 503 | 435 | 298 |
| 3.5 | 2 | 134 | 1 105 | 338 | 454 |
| 3.6 | 2 | 94 | 3 513 | 584 | 279 |
| 3.7 | 2 | 120 | 232 | 300 | 193 |
| 3.8 | 2 | 254 | fail | 188 | 487 |
| 3.9 | 2 | 122 | 300 | 371 | 391 |
| 3.10 | 2 | 118 | 633 | 672 | 234 |
| 3.11 | 4 | 166 | 6 654 | 1 662 | 710 |
| 3.12 | 5 | 258 | 7 649 | 2 945 | 1 059 |
| 3.15 | 6 | 1 039 | 3 059 | 5 544 | 4 286 |
| 3.16 | 10 | 896 | fail | 4 929 | 1 655 |
| 3.17 | 10 | fail | fail | 4 567 | fail |
| 3.18 | 12 | 1 036 | 24 698 | 9 308 | 4 862 |
| 3.19 | 20 | 6 277 | 7 243 | 5 541 | 4 243 |
| 3.20 | 20 | fail | 102376 | 5720 | fail |

In small-scale problems SolvOpt usually used less evaluations than any of the other solvers tested including our new solver while DGM was the most reliable solver tested (see Table 2). However, LDGBM usually used less evaluations than DGM and LMBM and the robustness of the solver was similar to SolvOpt and LMBM. Thus, we can say that our new solver is comparable with the existing solvers in small-scale settings.

Although SolvOpt solved small-scale problems both efficiently and robustly it was not in its element when solving the medium- and large-scale problems (see Table 3). The number of evaluations were still usually the smallest ones but the robustness was very weak.

Also our new solver LDGBM solved only five of the problems with 1000 variables with the desired accuracy (out of ten). However, the failures obtained with LDGBM were mostly inaccurate result: with the relaxed tolerance $\varepsilon = 10^{-2}$ LDGBM failed only in one problem (problem 2) while with SolvOpt the number of failures is still five even if the relaxed tolerance was used. This indicates that with LDGBM a better accuracy might be achieved with more tight stopping criterion.

In smaller problems ($n \leq 200$) the robustness of LDGBM was as good as that of DGM which, on the other hand, was the most robust of the existing solvers. LDGBM also used clearly less evaluations and computational time than DGM and LMBM.

Table 3: Summary of the results for medium and large-scale problems

| P | $n$ | SolvOpt<br>$nf/cpu$ | LMBM<br>$nf/cpu$ | DGM<br>$nf/cpu$ | LDGBM<br>$nf/cpu$ |
|---|---|---|---|---|---|
| 1 | 50 | 32 195/0.17 | 54 564/5.20 | 27 025/0.03 | 26 284/0.05 |
| 2 | 50 | 9 984/0.98 | fail | fail | fail |
| 3 | 50 | fail | 114 671/0.26 | 103 200/1.35 | 12 588/0.04 |
| 4 | 50 | fail | 97 701/1.09 | 89 616/1.86 | 13 858/0.17 |
| 5 | 50 | 2 237/0.03 | 84 240/0.81 | 36 083/0.27 | 6 548/0.07 |
| 6 | 50 | 854/0.01 | 106 807/0.74 | 9 201/0.04 | 3 969/0.02 |
| 7 | 50 | 3 728/0.13 | 127 804/3.05 | 41 548/1.09 | 17 760/0.47 |
| 8 | 50 | fail | fail | 92 527/0.95 | 32 915/0.08 |
| 9 | 50 | fail | 79 547/0.13 | 26 112/0.03 | 4 163/0.01 |
| 10 | 50 | fail | 169 377/0.40 | 52 670/0.74 | 16 177/0.04 |
| 1 | 200 | 555 779/9.23 | 848 449/6.54 | 312 544/0.59 | 598 321/2.35 |
| 2 | 200 | 48 770/79.86 | fail | 379 523/313.80 | fail |
| 3 | 200 | fail | 1 165 657/10.05 | 553 314/14.10 | 173 305/1.68 |
| 4 | 200 | fail | 503 113/22.24 | 568 910/32.67 | 104 501/4.38 |
| 5 | 200 | 12 889/0.78 | 46 957/1.79 | 261 919/7.02 | 22 860/0.91 |
| 6 | 200 | fail | 488 559/12.88 | fail | 31 825/0.73 |
| 7 | 200 | fail | 418 623/39.47 | 378 468/37.66 | 68 331/7.14 |
| 8 | 200 | fail | 325 371/1.63 | 862 819/17.09 | 80 260/0.53 |
| 9 | 200 | 19 063/0.32 | 313 324/1.99 | 163 043/0.59 | 12 540/0.09 |
| 10 | 200 | fail | 546 940/4.75 | 628 636/20.48 | 69 233/0.69 |
| 1 | 1000 | fail | fail | 10 592 047/92.43 | fail |
| 2 | 1000 | 44 213/1812.13 | fail | fail | fail |
| 3 | 1000 | fail | 4 910 632/208.60 | 3 579 840/111.49 | 1 484 246/61.77 |
| 4 | 1000 | fail | 1 314 594/282.92 | 3 085 985/452.59 | 762 840/165.82 |
| 5 | 1000 | 44 206/14.64 | 317 113/58.52 | 1 593 326/197.64 | 545 135/105.97 |
| 6 | 1000 | fail | 4 001 432/511.81 | 12 583 470/1107.24 | fail |
| 7 | 1000 | fail | fail | 4 455 499/1609.18 | fail |
| 8 | 1000 | fail | 2 302 536/55.69 | 2 993 617/54.78 | 182 975/4.67 |
| 9 | 1000 | fail | 820 998/25.62 | 993 300/16.12 | 70 181/2.17 |
| 10 | 1000 | fail | fail | fail | fail |

## 5 Conclusions

In this paper, we have described a new limited memory discrete gradient bundle method (LDGBM) for unconstrained nonsmooth optimization. LDGBM is a derivative free method and thus applicable to a broad class of optimization problems even if the (sub)gradient is not available. We have proved the global convergence of the method for locally Lipschitz continuous semismooth objective functions, which are not necessarily differentiable or convex.

The numerical experiments reported confirm that LDGBM is efficient for both convex and nonconvex nonsmooth optimization problems. With small-scale problems ($n \leq 20$) LDGBM was comparable with the existing solvers and with large numbers of variables it usually used clearly less function evaluations than the other solvers tested.

We can conclude that LDGBM is a good alternative to existing derivative free nonsmooth optimization algorithms and for medium- and large-scale problems it might well be the best choice.

## Acknowledgements

The work was financially supported by the University of Turku (Finland) and the University of Ballarat (Australia).

## References

[1] Audet, C., and Dennis, J. E. J. Analysis of generalized pattern searches. *SIAM Journal on Optimization 13* (2003), 889–903.

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

[3] Bagirov, A. M., and Ganjehlou, A. N. An approximate subgradient algorithm for unconstrained nonsmooth, nonconvex optimization. *Mathematical Methods of Operations Research 67* (2008), 187–206.

[4] Bagirov, A. M., Karasozen, B., and Sezer, M. Discrete gradient method: A derivative free method for nonsmooth optimization. *Journal of Optimization Theory and Applications 137* (2008), 317–334.

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

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

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

[8] 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.

[9] 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.

[10] 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.

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

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

[13] Custodio, A. L., Dennis, J. E., and Vicente, L. N. Using simplex gradients of nonsmooth functions in direct search methods. *IMA J. Numer. Anal. 28*, 4 (2008), 770–784.

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

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

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

[17] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1998.

[18] Goldstein, A. A. Optimization of Lipschitz continuous functions. *Matematical Programming 13*, 1 (1977), 14–22.

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

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

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

[24] Kappel, F., and Kuntsevich, A. An implementation of Shor's $r$-algorithm. *Computational Optimization and Applications 15* (2000), 193–205.

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

[26] Karmitsa, N., Bagirov, A., and Mäkelä, M. M. Comparing different nonsmooth optimization methods and software. *Optimization Methods and Software* (2011). to appear.

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

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

[29] Kuntsevich, A., and Kappel, F. SolvOpt — the solver for local nonlinear optimization problems. Karl-Franzens University of Graz: Graz, Austria, 1997.

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

[31] 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.

[32] 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.

[33] Lukšan, L., and Vlček, J. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, 2000.

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

[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. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization 15*, 6 (1977), 959–972.

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

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

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

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

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

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

[46] 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.