Bundle Algorithms for Nonsmooth Unconstrained Optimization

Vlček, Jan
1994

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Bundle Algorithms for Nonsmooth Unconstrained Optimization

Jan Vlček

Technical report No. 608

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 6605 3281 fax: (+422) 8585789
e-mail: uivt@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Bundle Algorithms for Nonsmooth Unconstrained Optimization

Jan Vlček [1]

Technical report No. 608

Abstract

We study mainly practical aspects of algorithms. Section 1 contains a brief introduction to Nonsmooth (Nondifferentiable) Optimization and a simple bundle algorithm. In Section 2 we incorporate the aggregation strategy, Section 3 contains a necessary generalization for the nonconvex case, in Section 4 we address the question of how to obtain the next iterate, Section 5 is devoted to a stopping criterion and in section 6 we give a technique for choosing the weight parameter. Some numerical experiments are reported in Section 7.

Keywords

Nondifferentiable minimization, numerical methods, descent methods

[1] This work was supported under grant No. 201/93/0429 given by the Czech Republic Grant Agency

1 Introduction

Consider the following nonlinear optimization problem

$$\text{minimize } f(x) \text{ on } R^n \tag{1.1}$$

Differently from the standard situation, we do not require f to be smooth. In sections 1 and 2 we suppose, that f is convex. In this case we can use the following substitution of the gradient: the *subdifferential* of f at x (see [1])

$$\partial f(x) = \{g \in R^n | \langle g, z - x \rangle \leq f(z) - f(x) \text{ for all } z \in R^n\} \tag{1.2}$$

This $\partial(x)$ is a nonempty, convex and compact set, which shrinks to the gradient, whenever f is differentiable at x; the elements of $\partial(x)$ are called *subgradients*. We make the general assumption: at every x, we know $f(x)$ and one (arbitrary) $g \in \partial f(x)$.

The methods for nonsmooth optimization can be divided into two main classes: subgradient methods and bundle methods; we restrict ourselves to the latter, which seem to be the most promising.

A long-known algorithm to minimize convex f is the *cutting plane* algorithm (see [2]-[3]): At step k, let the iterates $y_1, \ldots, y_k$ have been generated, together with the corresponding function values $f(y_1), \ldots, f(y_k)$ and subgradients $g_1 \in \partial f(y_1), \ldots, g_k \in \partial f(y_k)$. We define the *cutting plane approximation* of f (associated with above *bundle* of information) to be the piecewise linear function

$$\hat{f}_k(x) = \max\{f(y_i) + \langle g_i, x - y_i \rangle : \quad i = 1, \ldots, k\} \tag{1.3}$$

It results immediately from (1.2) that

$$\hat{f}_k(x) \leq f(x) \quad \text{and} \quad \hat{f}_k(y_i) = f(y_i), \quad i = 1, \ldots, k. \tag{1.4}$$

Minimization of this cutting plane model in some convex compact set, which contains an optimal point, provides the next iterate y_{k+1}.

Unfortunately, the numerical behaviour of this algorithm is extremely poor. Therefore some stabilizing techniques were proposed. With respect to an available routine for solving a quadratic programming problem (see [4]) we choose the following process (see [1], also for its convergence properties) as our basic algorithm, which is in comparison with [1] slightly adapted with regard to next sections (e.g. in the quadratic programming subproblem we use equivalently $f_i^k - f(x_k)$ instead of f_i^k) :

Algorithm 1.1.

Step 0: *(Initialization).* Select a starting point $x_1 \in R^n$ and set $k = 1$, $y_1 = x_1$. Choose a weight (penalty parameter) $u > 0$ and line search parameter $m_L \in (0, 1)$; compute $f(y_1)$ and $g_1 \in \partial f(y_1)$.

Step 1: *(Direction finding).* Find the solution (d_k, v_k) to the k-th quadratic programming subproblem

minimize $\quad v + \frac{u}{2}|d|^2$ over all $(d, v) \in R^n \times R$

satisfying $\quad f_i^k - f(x_k) + \langle g_i, d\rangle \leq v$ for $i = 1, \ldots, k$, where

$$f_i^k = f(y_i) + \langle g_i, x_k - y_i\rangle \leq f(x_k). \tag{1.5}$$

Step 2: *(Stopping criterion).* If x_k is "close enough" to the required solution, then terminate.

Step 3: *(Line search).* Set $y_{k+1} = x_k + d_k$ and compute $f(y_{k+1})$ and $g_{k+1} \in \partial f(y_{k+1})$. If

$$f(y_{k+1}) \leq \ f(x_k) - m_L\delta_k,$$

where

$$\delta_k = f(x_k) - \hat{f}(y_{k+1}) - \frac{1}{2}u|d_k|^2 \geq 0,$$

set $x_{k+1} = y_{k+1}$ (*descent* or *serious step*); otherwise set $x_{k+1} = x_k$ (*null step*).

Step 4: *(Updating).* Replace k by $k + 1$ and go to *Step 1.*

Note that this is only the crude form; e.g. we do not need store the auxiliary points y_i, since $f_i^{k+1} - f_i^k = \langle g_i, x_{k+1} - x_k\rangle$, so we have the following recursive updating formula:

$$f_i^{k+1} = f_i^k + \langle g_i, x_{k+1} - x_k\rangle, \quad i = 1, \ldots, k \tag{1.6}$$

Recomended values of constants are proposed in Section 7. In next sections we give some modifications and refinements.

2 Subgradient Aggregation

There is the great drawback of the above algorithm: the number of constraints (in *Step 1*) and stored subgradients grows unboundedly, since it is equal to the number of iterations. Therefore we modify it using the *subgradient aggregation* strategy (see [5], also for its convergence properties). The idea is to aggregate the constraints generated by the past subgradients.

Algorithm 2.1.

Step 0: *(Initialization).* Select a starting point $x_1 \in R^n$, a final accuracy tolerance $\varepsilon \geq 0$ and a line search parameter $m_L \in (0, 1)$. Set $y_1 = x_1$, $J_1 = \{1\}$ and compute $f(y_1)$, $g_1 \in \partial f(y_1)$. Set $p_0 = g_1$, $f_p^1 = f_1^1 = f(y_1)$ and the counter $k = 1$.

Step 1: *(Direction finding).* Find the solution (d_k, v_k) to the k-th quadratic programming subproblem

$$\begin{array}{ll} \text{minimize} & v + \frac{1}{2}|d|^2 \text{ over all } (d,v) \in R^n \times R \\ \text{subject to} & f_j^k - f(x_k) + \langle g_j, d\rangle \leq v \text{ for } j \in J_k, \end{array}$$

$$f_p^k - f(x_k) + \langle p_{k-1}, d\rangle \leq v$$

which is by duality equivalent to finding multipliers λ_j^k, $j \in J_k$, and λ_p^k that solve the k-th dual subproblem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left|\sum_{j\in J_k} \lambda_j g_j + \lambda_p p_{k-1}\right|^2 - \sum_{j\in J_k} \lambda_j f_j^k - \lambda_p f_p^k \\ \text{subject to} & \lambda_j \geq 0,\ j \in J_k,\ \lambda_p \geq 0,\ \sum_{j\in J_k} \lambda_j + \lambda_p = 1. \end{array}$$

Compute the aggregate couple

$$(p_k, \tilde{f}_p^k) = \sum_{j\in J_k} \lambda_j^k (g_j, f_j^k) + \lambda_p^k (p_{k-1}, f_p^k)$$

and set $d_k = -p_k$, $v_k = -\{|p_k|^2 + f(x_k) - \tilde{f}_p^k\}$, where $v_k \leq 0$ is the same as in the primal subproblem.

Step 2: *(Stopping criterion).* If $\frac{1}{2}|p_k|^2 + f(x_k) - \tilde{f}_p^k \leq \varepsilon$ then stop.

Step 3: *(Line search).* Set $y_{k+1} = x_k + d_k$ and evaluate $f(y_{k+1})$ and $g_{k+1} \in \partial f(y_{k+1})$. If

$$f(y_{k+1}) \leq f(x_k) + m_L v_k,$$

set $x_{k+1} = y_{k+1}$ (*descent* or *serious step*), otherwise set $x_{k+1} = x_k$ (*null step*).

Step 4: *(Updating).* Choose a set $\hat{J}_k \subset \{1, \ldots, k\}$ and calculate

$$\begin{aligned} f_i^{k+1} &= f_i^k + \langle g_i, x_{k+1} - x_k\rangle, \quad i \in \hat{J}_k, \\ f_{k+1}^{k+1} &= f(y_{k+1}) + \langle g_{k+1}, x_{k+1} - y_{k+1}\rangle, \\ f_p^{k+1} &= \tilde{f}_p^k + \langle p_k, x_{k+1} - x_k\rangle. \end{aligned}$$

Set $J_{k+1} = \hat{J}_k \cup \{k+1\}$, replace k by $k+1$ and go to *Step 1.*

Note that here it is $u = 1$ in comparison with the Algorithm 1.1. We shall return to the weight u in Section 4.

The index set $\hat{J}_k$ may be determined arbitrarily. I have tested some techniques for choosing it, but without any apparent effect. In the present version it is $\hat{J}_k = \{\max(k - M, 0) + 1, \ldots, k\}$, where $M \geq 2$ is a user-supplied bound on the number of indices.

3 Nonconvexity

First we note that a nonconvex function may have several local minima, thus we content ourselves with finding only one of them.

It is possible to generalize the subdifferential to nonconvex locally Lipschitz functions (see [6]), but it does not possess the property (1.2). Thus the inequality in (1.4) does not hold and it is necessary to introduce so-called *subgradient locality measures* (see [5]) for weighting past subgradients at search direction finding. To define them we need some upper estimate of $|x_k - y_j|$, which does not require the storage of points y_j, e.g.

$$s_j^k = |x_j - y_j| + \sum_{i=j}^{k-1} |x_{i+1} - x_i| \quad \text{for} \quad j < k, \;\; s_k^k = |x_k - y_k|;$$

we can use the following recursive updating formula:

$$s_j^{k+1} = s_j^k + |x_{k+1} - x_k|, \quad j = 1, \ldots, k.$$

Denoting

$$\alpha_j^k = \max\{|f_j^k - f(x_k)|, \gamma(s_j^k)^2\}, \quad \gamma \geq 0,$$

we shall call s_j^k the j-th *distance measure* and α_j^k the j-th *subgradient locality measure* at the k-th iteration with the convention that $\gamma = 0$ if f is convex (then $-\alpha_j^k = f_j^k - f(x_k)$), and $\gamma > 0$ in the nonconvex case.

A generalized algorithm to the nonconvex case will be given in the next section.

4 Line Search

The hardest problem, which is created by nonsmoothness, consists in the fact that the direction opposite to an arbitrary subgradient need not be one of descent. This difficulty forces us also to modify the classical line search operations. Standard concepts in bundle methods are a *serious* (or *descent*) *step* and a *null step*; in [6] there are two serious steps - *long* and *short*.

Note that more important than to achieve a sufficient decrease of the objective function is to add the new subgradient information to the bundle to generate a better next search direction (e.g. when in the algorithms above in *Step 3* we try to reach a decrease of $f(x)$ instead of performing the null step, it may cause a significant increasing of the number of iterations).

We give now a simplified algorithm after [6].

Algorithm 4.1

Step 0: *(Initialization).* Select a starting point $x_1 \in R^n$, a final accuracy tolerance $\varepsilon \geq 0$, a weight $u > 0$, the maximum number of stored subgradients $M \geq 2$, the distance measure parameter $\gamma > 0$ ($\gamma = 0$ if f is convex) and line search parameters $m_L \in (0,1)$, $m_R \in (m_L, 1)$ and a lower bound for long serious steps $\overline{t} \in (0,1]$. Set $y_1 = x_1$, $J_1 = \{1\}$ and compute $f(y_1)$, $g_1 \in \partial f(y_1)$. Set the counter $k = 1$, the line search parameter $\zeta = 1 - .5/(1 - m_L)$ and initialize $p_0 = g_1$, $f_p^1 = f_1^1 = f(y_1)$, $s_p^1 = s_1^1 = 0$.

Step 1: *(Direction finding).* Find the solution $(d_k, \overline{v}_k)$ to the k-th quadratic programming subproblem

$$\begin{array}{ll} \text{minimize} & \overline{v} + \frac{u}{2}|d|^2 \text{ over all } (d, \overline{v}) \in R^n \times R \\ \text{subject to} & -\alpha_j^k + \langle g_j, d\rangle \leq \overline{v} \text{ for } j \in J_k,\ -\alpha_p^k + \langle p_{k-1}, d\rangle \leq \overline{v}, \end{array}$$

where

$$\begin{aligned} \alpha_j^k &= \max\{|f_j^k - f(x_k)|, \gamma(s_j^k)^2\} \quad \text{for} \quad j \in J_k, \\ \alpha_p^k &= \max\{|f_p^k - f(x_k)|, \gamma(s_p^k)^2\}, \end{aligned}$$

which is by duality equivalent to finding multipliers λ_j^k, $j \in J_k$, and λ_p^k that solve the k-th dual subproblem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2u}\left|\sum_{j\in J_k} \lambda_j g_j + \lambda_p p_{k-1}\right|^2 + \sum_{j\in J_k} \lambda_j \alpha_j^k + \lambda_p \alpha_p^k \\ \text{subject to} & \lambda_j \geq 0,\ j \in J_k,\ \lambda_p \geq 0,\ \sum\limits_{j\in J_k} \lambda_j + \lambda_p = 1, \end{array}$$

with $\overline{v}_k = -\left\{\frac{1}{u}|p_k|^2 + \sum\limits_{j\in J_k} \lambda_j^k \alpha_j^k + \lambda_p^k \alpha_p^k\right\}$.

Set

$$(p_k, \tilde{f}_p^k, \tilde{s}_p^k) = \sum_{j\in J_k} \lambda_j^k (g_j, f_j^k, s_j^k) + \lambda_p^k (p_{k-1}, f_p^k, s_p^k),$$

$$\tilde{\alpha}_p^k = \max\{|\tilde{f}_p^k - f(x_k)|, \gamma(\tilde{s}_p^k)^2\},\ v_k = -\frac{1}{u}|p_k|^2 - \tilde{\alpha}_p^k,\ d_k = -\frac{1}{u}p_k.$$

Step 2: *(Stopping criterion).* If $\frac{1}{2}|p_k|^2 + \tilde{\alpha}_p^k \leq \varepsilon$ then stop.

Step 3: *(Line search).* By line search Algorithm 4.2 find step sizes t_L^k, t_R^k, set $x_{k+1} = x_k + t_L^k d_k$, $y_{k+1} = x_k + t_R^k d_k$, and evaluate $f(y_{k+1})$ and $g_{k+1} \in \partial f(y_{k+1})$.

Step 4: *(Updating).* Calculate the linearization values

$$\begin{aligned}
f_i^{k+1} &= f_i^k + \langle g_i, x_{k+1} - x_k \rangle, \quad i \in J_k, \\
f_{k+1}^{k+1} &= f(y_{k+1}) + \langle g_{k+1}, x_{k+1} - y_{k+1} \rangle, \\
f_p^{k+1} &= \tilde{f}_p^k + \langle p_k, x_{k+1} - x_k \rangle, \\
s_i^{k+1} &= s_i^k + t_L^k |d_k|, \quad i \in J_k,\ s_{k+1}^{k+1} = (t_R^k - t_L^k)|d_k|, \\
s_p^{k+1} &= \tilde{s}_p^k + t_L^k |d_k|.
\end{aligned}$$

Set $J_{k+1} = J_k \cup \{k+1\}$ and increase k by 1; if card $J_k > M$, then $J_k = J_k \backslash \{\min j | j \in J_k\}$. Go to *Step 1.*

Algorithm 4.2 (Line search)

Step i: Set $t_L^k = 0$ and $t = t_U = 1$.

Step ii: If $f(x_k + td_k) \leq f(x_k) + m_L t v_k$, set $t_L^k = t$, otherwise set $t_U = t$.

Step iii: If $t_L^k \geq \overline{t}$ set $t_R^k = t_L^k$ and stop (*long serious step*), otherwise calculate $g \in \partial f(x_k + td_k)$ and

$$\beta = \max\{|f(x_k + t_L^k d_k) - f(x_k + td_k) + (t - t_L^k) g^T d_k|,\ \gamma (t - t_L^k)^2 |d_k|^2\}.$$

Then if $-\beta + g^T d_k \geq m_R v_k$, set $t_R^k = t$ and stop (a *null step* if $t_L^k = 0$, a *short serious step* otherwise).

Step iv: If $t_L^k = 0$ then set

$$t = \max\{\zeta \cdot t_U, \frac{1}{2} t_U^2 v_k / (t_U v_k + f(x_k) - f(x_k + t_U d_k)\}$$

and if $t_L^k > 0$ then set $t = \frac{1}{2}(t_L^k + t_U)$. Go to *Step (ii).*

In the course of computational testing some drawbacks appeared:

- on some problems there was too many null steps, particularly at first iterations;
- at the first iteration, the inequality in *Step (ii)* was not satisfied in some cases, in spite of the significant decrease of the objective function;
- the minimal value of $f(x)$ could be uselessly forgotten.

Therefore and with respect to an available quadratic interpolation routine I have modified the line search procedure as follows:

Algorithm 4.3 (Line search)

Initialization: If $k = 1$ set $i_n = 0$ (number of consecutive null steps).

Step i: Set $t_L^k = t_P = 0$ and $t = t_U = 1$.

Step ii: If $f(x_k + td_k) \leq f(x_k) + \max\{m_L t v_k, -1\}$, set $t_P = t$, otherwise set $t_U = t$. If $f(x_k + td_k) < f(x_k)$, set $t_L^k = t$.

Step iii: If $t_P > 0$ and $t_L^k \geq \overline{t}$, set $t_R^k = t_L^k$, $i_n = 0$ and stop (*serious step*), otherwise calculate $g \in \partial f(x_k + td_k)$ and

$$\beta = \max\{|f(x_k + t_L^k d_k) - f(x_k + td_k) + (t - t_L^k)g^T d_k|,\ \gamma(t - t_L^k)^2|d_k|^2\}.$$

Then if $(-\beta + g^T d_k \geq m_R v_k$ and $(t < 1$ or $(k > 1$ and $i_n \leq 6)))$ or $t_U - t_P < \overline{t}$, set $t_R^k = t$ and ($i_n = i_n + 1$ if $t_L^k = 0$, $i_n = 0$ otherwise) and stop (a *null step* if $t_L^k = 0$, a *short step* otherwise).

Step iv: If $t_P = 0$ then set

$$t = \min\{0.9t_U, \max\{t_U/100, \frac{1}{2}t_U^2 v_k/(t_U v_k + f(x_k) - f(x_k + t_U d_k)\}\}$$

and if $t_P > 0$ then set $t = \frac{1}{2}(t_P + t_U)$. Go to *Step (ii)*.

5 Stopping Criterion

To balance a precision of the objective function, I have modified *Step 2* in the Algorithm 4.1 as follows:

Step 2': *(Stopping criterion).* If

$$\frac{1}{2}|p_k|^2 + 500 \cdot \max\{|\tilde{f}_p^k - f(x_k)|, (\tilde{s}_p^k)^2\}/(|f(x_k)| + 0.001) \leq \varepsilon,$$

or $|f(x_k) - f(x_{k-1})| \leq \varepsilon_f$ in two consecutive iterations, where $\varepsilon_f > 0$, then stop.

6 Weight Updating

In the algorithm given in [6] the weight parameter u is not constant, but it is used a following safeguard quadratic interpolation technique by **Kiwiel** (see [7]) for updating u_k (it can be placed in the end of *Step 4* of the Algorithm 4.1):

Algorithm 6.1 (Weight updating)

Initialization: If $k = 1$ set $\varepsilon_v^k = +\infty$, $i_u^k = 0$ and select a lower bound for weights $u_{\min} > 0$.

Step (a) Set $u = u_k$. If $y_{k+1} \not\equiv x_k + d_k$ then exit (see below).

Step (b) If $x_{k+1} = x_k$ go to *Step (f)*.

Step (c) If $f(y_{k+1}) \leq f(x_k) + m_R v_k$ and if $i_u^k > 0$ then set $u = 2u_k(1 - [f(y_{k+1}) - f(x_k)]/v_k)$ and go to *Step (e)*.

Step (d) If $i_u^k > 3$ set $u = u_k/2$.

Step (e) Set $u_{k+1} = \max\{u, u_k/10, u_{\min}\}$, $\varepsilon_v^{k+1} = \max\{\varepsilon_v^k, -2v_k\}$ and $i_u^{k+1} = \max\{i_u^k + 1, 1\}$. If $u_{k+1} \neq u_k$ set $i_u^{k+1} = 1$. Exit.

Step (f) Set $\varepsilon_v^{k+1} = \min\{\varepsilon_v^k, |p_k| + \tilde{\alpha}_p^k\}$. If $\alpha_{k+1}^{k+1} > \max\{\varepsilon_v^{k+1}, -10v_k\}$ and $i_u^k < -3$, set $u = 2u_k(1 - [f(y_{k+1}) - f(x_k)]/v_k)$. Set $u_{k+1} = \min\{u, 10u_k\}$ and $i_u^{k+1} = \min\{i_u^k - 1, -1\}$. If $u_{k+1} \neq u_k$ set $i_u^{k+1} = -1$. Exit.

This procedure is not quite corresponding to the used Algorithm 4.2 (or 4.3), since

- the given quadratic interpolation formula is derived only for unit step lengths ($y_{k+1} = x_k + d_k$),
- the weight updating algorithm is sensitive to the definition of null step; if we use short (serious) steps or interpolation instead of null steps, it is possible that there is not enough consecutive null steps and thus u_k is never increased.

Moreover I have found that the change of u_k should not be damped so much - after numerical experiments it seems that u_k should jump at the boundary of regions inside which the gradient ∇f exists and is continuous. Therefore I propose the following procedure, where the quadratic interpolation is generalized to any step lengths and therefore is satisfactory also for null and short steps.

Algorithm 6.2 (Line search + weight updating)

Initialization: If $k = 1$ set $u_1 = 1$, $i_n = 0$ (number of consecutive null steps) and select a lower bound for weights $u_{\min} > 0$.

(i): Set $t_L^k = t_P = 0$ and $t = t_U = 1$.

(ii): Set $t_P = t$ if $f(x_k + td_k) \leq f(x_k) + \max\{m_L t v_k, -1\}$, $t_U = t$ otherwise.
If $f(x_k + td_k) < f(x_k)$ set $t_L^k = t$.

(iii): Calculate $g \in \partial f(x_k + td_k)$.
Set $u_p = 2(tg^T d_k + f(x_k) - f(x_k + td_k))/(t|d_k|)^2$.
If $t < 1$ set

$$u_q = 2(f(x_k + t_U d_k) - f(x_k + td_k) - (t_U - t)g^T d_k)/((t_U - t)|d_k|)^2,$$

otherwise set

$$\begin{aligned} u_q &= 2(f(x_k + td_k) - f(x_k) - tg_k^T d_k)/(t|d_k|)^2 \text{ if } i_n = 0, \\ u_q &= -1 \ otherwise. \end{aligned}$$

If $u_p < -u_{\min}$ set $u_p = u_q$, otherwise if $u_q \geq -u_{\min}$ set $u_p = \min\{u_p, u_q\}$.

(iv): If $t_P > 0$ and $t_L^k \geq \overline{t}$, set $t_R^k = t_L^k$, $i_n = 0$ and go to *(vi)* (*serious step*), otherwise calculate

$$\beta = \max\{|f(x_k + t_L^k d_k) - f(x_k + td_k) + (t - t_L^k)g^T d_k|, \gamma(t - t_L^k)^2 |d_k|^2\}.$$

Then if

$$(-\beta + g^T d_k \geq m_R v_k \text{ and } (t < 1 \text{ or } (k > 1 \text{ and } i_n \leq 6))) \text{ or } t_U - t_P < \overline{t},$$

set $t_R^k = t$ and ($i_n = i_n + 1$ if $t_L^k = 0$, $i_n = 0$ if $t_L^k = t_R^k$, $i_n = -1$ otherwise) and go to *(vi)* (a *null step* if $t_L^k = 0$, a *short step* otherwise).

(v): If $t_P = 0$ then set

$$t = \min\{0.9t_U, \max\{t_U/100, \frac{1}{2}t_U^2 v_k/(t_U v_k + f(x_k) - f(x_k + t_U d_k)\}\}$$

and if $t_P > 0$ then set $t = \frac{1}{2}(t_P + t_U)$. Go to *Step (ii)*.

(vi): If $-u_{\min}/10 \leq u_p \leq 100/u_{\min}$ and ($i_n = 0$ or $i_n \geq 5$ or $f(x_k + d_k) \leq f(x_k) + \max\{m_L v_k, -1\}$), then set

$$u_{k+1} = \min\{\max\{u_p, u_{\min}, u_k/10\}, 1/u_{\min}, 10u_k\}.$$

Stop.

7 Numerical Experiments

In this section we compare our results for 22 standard examples from the literature with those obtained by the BT algorithm (nonconvex version) by H. Schramm. Problems 1-16 are described in [6], problems 17-18 in [8], problems 19-20 in [9], problems 21, 22 respectively in [10], [11]. In the table 1 we give optimal values of tested functions for the problems.

Nr.	Problem	Minimum	Nr.	Problem	Minimum
1	Rosenbrock	0.0	12	Maxquad1	-0.84140833
2	Crescent	0.0	13	Maxq	0.0
3	CB2	1.9522245	14	Maxl	0.0
4	CB3	2.0	15	TR48	-638565.0
5	DEM	-3.0	16	Goffin	0.0
6	QL	7.20	17	El Attar	0.55981306
7	LQ	-1.4142136	18	Wolfe	-8.0
8	Mifflin1	-1.0	19	MXHILB	0.0
9	Mifflin2	-1.0	20	L1HILB	0.0
10	Rosen	-44.0	21	Gill	9.7857721
11	Shor	22.600162	22	Steiner2	16.703838

Table 1: Optimal values for problems

We present the results of two algorithms. The first one A1 implements the Algorithm 4.1, where the line search Algorithm 4.2 is replaced by the Algorithm 4.3 with constant weight parameter $u = 1$, the second Algorithm A2 implements a combination of the Algorithms 4.1 and 6.2.

The parameters have the following values: $m_L = 0.01$, $m_R = 0.5$, $\overline{t} = 0.001$, $\gamma = 0.25$, $u_{\min} = 0.002$, $\varepsilon = 10^{-6}$, $\varepsilon_f = 10^{-8}$.

Tables 2, 3 respectively contains results for the choice $M = n + 3$, $M = 106$ (**ni** is a number of iterations, **nf** is a number of function and subgradient evaluations, f^* is a final value of the objective function).

As a conclusion from our limited numerical experiments we may state that

- our method is comparable with the BT algorithm;
- the version A2 requires in many examples less function evaluations than A1;
- the number of function evaluations for $M = 106$ is mostly not much smaller than for $M = n + 3$ and therefore the bundle size $(n + 3)$ seems to be satisfactory.

Alg.	**BT**			**A1**			**A2**		
Nr.	**ni**	**nf**	f^*	**ni**	**nf**	f^*	**ni**	**nf**	f^*
1	92	104	.1159E-10	59	68	.2869E-08	39	42	.4061E-09
2	14	17	.00001170	12	14	.3933E-11	18	20	.4045E-15
3	24	26	1.9522245	33	35	1.9522245	32	34	1.9522245
4	13	15	2.0000002	42	47	2.0000000	14	16	2.0000000
5	12	17	-3.0000000	15	17	-3.0000000	17	19	-3.0000000
6	17	18	7.2000000	26	28	7.2000000	21	23	7.2000023
7	6	12	-1.4142135	10	11	-1.4142136	11	12	-1.4142136
8	62	69	-1.0000000	35	37	-1.0000000	30	32	-1.0000000
9	27	32	-1.0000000	8	10	-1.0000000	13	15	-1.0000000
10	60	65	-44.000000	73	75	-44.000000	43	45	-43.999999
11	35	36	22.600163	90	92	22.600162	27	29	22.600162
12	75	83	-.84140833	223	229	-.84140833	61	62	-.84140832
13	204	205	.2429E-11	329	330	.4186E-06	343	344	.9049E-07
14	91	93	.3442E-14	261	262	.00028127	118	119	.05810643
15	500	570	-638346.40	500	508	-638357.22	500	504	-632141.59
16	169	172	.1546E-13	304	332	.00074720	400	422	.3275E-15
17	95	105	.55981991	131	135	.55981307	104	110	.55981457
18	23	30	-7.9999999	59	63	-8.0000000	45	48	-8.0000000
19	206	256	.4400E-06	132	133	.00636343	142	227	.3688E-05
20	170	172	.1937E-08	381	382	.6995E-06	258	280	.1584E-04
21	500	541	9.7857861	431	467	9.7857721	254	257	9.7857748
22	226	228	16.703838	147	159	16.703839	151	164	16.703876
$\sum$	2621	2866		3301	3434		2641	2824	
Time			9:03.76			2:24.67			2:35.82

Table 2: Results for $M = n + 3$

Alg.	BT			A1			A2		
Nr.	ni	nf	f^*	ni	nf	f^*	ni	nf	f^*
1	91	101	.1602E-09	49	56	.5344E-09	92	96	.2578E-06
2	14	17	.1170E-04	12	14	.3933E-11	13	15	.4439E-11
3	24	26	1.9522245	25	27	1.9522245	26	28	1.9522249
4	13	15	2.0000002	33	37	2.0000000	14	16	2.0000000
5	12	17	-3.0000000	15	17	-3.0000000	17	19	-3.0000000
6	17	18	7.2000000	25	27	7.2000000	25	27	7.2000031
7	6	12	-1.4142135	10	11	-1.4142136	12	13	-1.4142136
8	58	66	-1.0000000	34	36	-1.0000000	17	19	-1.0000000
9	32	37	-1.0000000	8	10	-1.0000000	13	15	-1.0000000
10	60	65	-44.000000	59	61	-44.000000	44	46	-43.999997
11	53	54	22.600162	65	67	22.600162	27	29	22.600162
12	98	105	-.84140557	200	203	-.84140833	61	62	-.84140832
13	141	142	.9898E-12	329	330	.4186E-06	343	344	.9049E-07
14	74	76	.1277E-14	261	262	.2813E-03	118	119	.05810643
15	500	556	-638340.33	500	508	-638354.98	500	503	-634129.98
16	169	172	.5608E-14	240	254	? 1.935658	324	333	.4997E-15
17	72	85	? 1.086660	106	107	.55981308	112	113	.55981625
18	23	30	-7.9999999	42	46	-8.0000000	43	46	-8.0000000
19	177	204	.3379E-06	132	133	.6363E-02	142	227	.3688E-05
20	176	179	.1758E-08	381	382	.6995E-06	301	307	.1928E-04
21	93	100	? 11.86699	500	522	9.7892591	500	502	9.7858202
22	101	103	16.703838	128	133	16.703838	162	170	16.703839
$\sum$	2004	2180		3154	3243		2906	3049	
Time			1:40:26.37			2:55.43			3:01.47

Table 3: Results for $M = 106$

Bibliography

[1] Lemaréchal C. and Zowe J., *A Condensed Introduction to Bundle Methods in Nonsmooth Optimization*, in "Algorithms for Continuous Optimization" (E. Spedicato ed.). Kluwer Academic Publishers, 1994.

[2] Cheney E.W. and Goldstein A.A., *Newton's Method for Convex Programming and Tchebycheff Approximation*, Numer. Math. **1** (1959), 253-268.

[3] Kelley J.E., *The Cutting Plane Method for Solving Convex Programs*, SIAM J. **8** (1960), 703-712.

[4] Lukšan L.: *Dual Method for Solving a Special Problem of Quadratic Programming as a Subproblem at Linearly Constrained Nonlinear Minimax Approximation*, Kybernetika **20** (1984), 6, 445-457.

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

[6] Mäkelä M.M. and Neittaanmäki, "Nonsmooth Optimization", World Scientific Publishing Co. Pte. Ltd., 1992.

[7] Kiwiel K.C., *Proximity Control in Bundle Methods for Convex Nondifferentiable Minimization*, Mathematical Programming **46** (1980), 105-122.

[8] Zowe J., *Nondifferentiable Optimization,* in "Computational Mathematical Programming", 1985, pp. 323-356.

[9] Kiwiel K.C., *An Ellipsoid Trust Region Bundle Method for Nonsmooth Convex Minimization*, SIAM Journal on Control and Optimization **27** (1989), 737-757.

[10] Bihain A., *Optimization of Upper Semidifferentiable Functions,* Journal of Optimization Theory and Applications **4** (1984), 545-568.

[11] Facchinei F. and Lucidi S., *Nonmonotone Bundle-Type Scheme for Convex Nonsmooth Minimization,* Journal of Optimization Theory and Applications **76** (1993), 241-257.