EngOpt 2008 - International Conference on Engineering Optimization
Rio de Janeiro, Brazil, 01 - 05 June 2008.

Limited Memory Bundle Method for Large Bound Constrained Nonsmooth Optimization

Napsu Karmitsa

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

1. Abstract
Practical optimization problems often involve nonsmooth functions of hundreds or thousands of variables. As a rule, the variables in such large problems are restricted to certain meaningful intervals. In the report [Haarala, Mäkelä, 2006] we have described an efficient adaptive limited memory bundle method for large-scale nonsmooth, possibly nonconvex, bound constrained optimization. Although it works very well in numerical experiments it suffers from one theoretical drawback. Namely, it is not necessarily globally convergent. In this paper, a globally convergent variant of this method is proposed. In addition, some results from numerical experiments are given.
2. Keywords: Nondifferentiable programming, large-scale optimization, bundle methods, limited memory methods, box constraints.

3. Introduction

In this paper, we describe an adaptive limited memory bundle algorithm (LMBM-B) for solving large, possibly nonconvex, nonsmooth (nondifferentiable) bound constrained optimization problems. We write this problem as

$$\begin{cases} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x}^l \leq \boldsymbol{x} \leq \boldsymbol{x}^u, \end{cases} \tag{1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is supposed to be locally Lipschitz continuous and the number of variables n is supposed to be large (say 1000 or more). Moreover, the vectors $\boldsymbol{x}^l$ and $\boldsymbol{x}^u$ representing the lower and the upper bounds on the variables are fixed and the inequalities in Eq.(1) are taken component-wise. A point $\boldsymbol{x} \in \mathbb{R}^n$ satisfying the bounds is called *feasible* and a set $\mathcal{F}$ of all feasible points is called the *feasible region* for problem (1). That is, $\mathcal{F} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^l \leq \boldsymbol{x} \leq \boldsymbol{x}^u\}$.

In [1, 2, 3] we have proposed a limited memory bundle method (LMBM) for general, possibly nonconvex, nonsmooth large-scale unconstrained optimization. LMBM is a hybrid of the variable metric bundle methods [4, 5] and the limited memory variable metric methods (see e.g. [6, 7]), where the first ones have been developed for small- and medium-scale nonsmooth optimization and the latter ones, on the contrary, for smooth large-scale optimization. LMBM exploits the ideas of the variable metric bundle methods, namely the utilization of null steps and simple aggregation of subgradients (generalized gradients [8]), but the search direction is calculated using a limited memory approach. Therefore, the time-consuming quadratic direction finding problem appearing in standard bundle methods (see e.g. [9, 10, 11]) need not to be solved and the number of stored subgradients (i.e. the size of the bundle) is independent of the dimension of the problem. Furthermore, LMBM uses only few vectors to represent the variable metric updates and, thus, it avoids storing and manipulating large matrices as is the case in variable metric bundle methods [4, 5].

In [12], a new variant of the method, LMBM-B, suitable for solving bound constrained problems was introduced. The constraint handling in LMBM-B is based on gradient projection (naturally, we use subgradients instead of gradients) and dual subspace minimization and it is adopted from the smooth limited memory BFGS method for bound constrained optimization [13]. Although numerically very efficient, the method described in [12] is not necessary globally convergent in nonsmooth case.

In order to prove the global convergence of LMBM-B some modifications had to be made. Namely, we included stark projections in aggregation procedure to guarantee the convergence of aggregate subgradients to zero, some corrections to limited memory matrices to preserve sufficient positive definiteness and boundedness of these matrices whenever necessary, and a slightly modified line search procedure. Although this may sound like an easy task several open question had to be answered and many implementational challenges had to be solved before preserving even a hint of the efficiency of the previous version together with the theoretical convergence properties.

In what follows, we assume that at every feasible point $\boldsymbol{x} \in \mathcal{F}$ we can evaluate the value of the objective function $f(\boldsymbol{x})$ and an arbitrary subgradient $\boldsymbol{\xi} \in \mathbb{R}^n$ from the subdifferential [8]

$$\partial f(\boldsymbol{x}) = \text{conv}\{ \lim_{i \to \infty} \nabla f(\boldsymbol{x}_i) \mid \boldsymbol{x}_i \to \boldsymbol{x} \text{ and } \nabla f(\boldsymbol{x}_i) \text{ exists} \}, \tag{2}$$

where "conv" denotes the convex hull of a set.

4. Method

The globally convergent version of LMBM-B (see Figure 1) is characterized by the usage of null steps together with the aggregation and projection of subgradients. Moreover, the limited memory approach is utilized in the calculation of the search direction and the aggregate values. The usage of null steps gives further information about the nonsmooth objective function in the case the search direction is not "good enough". On the other hand, simple aggregation and stark projection of subgradients guarantees the convergence of projected aggregate subgradients to zero and make it possible to evaluate a termination criterion.

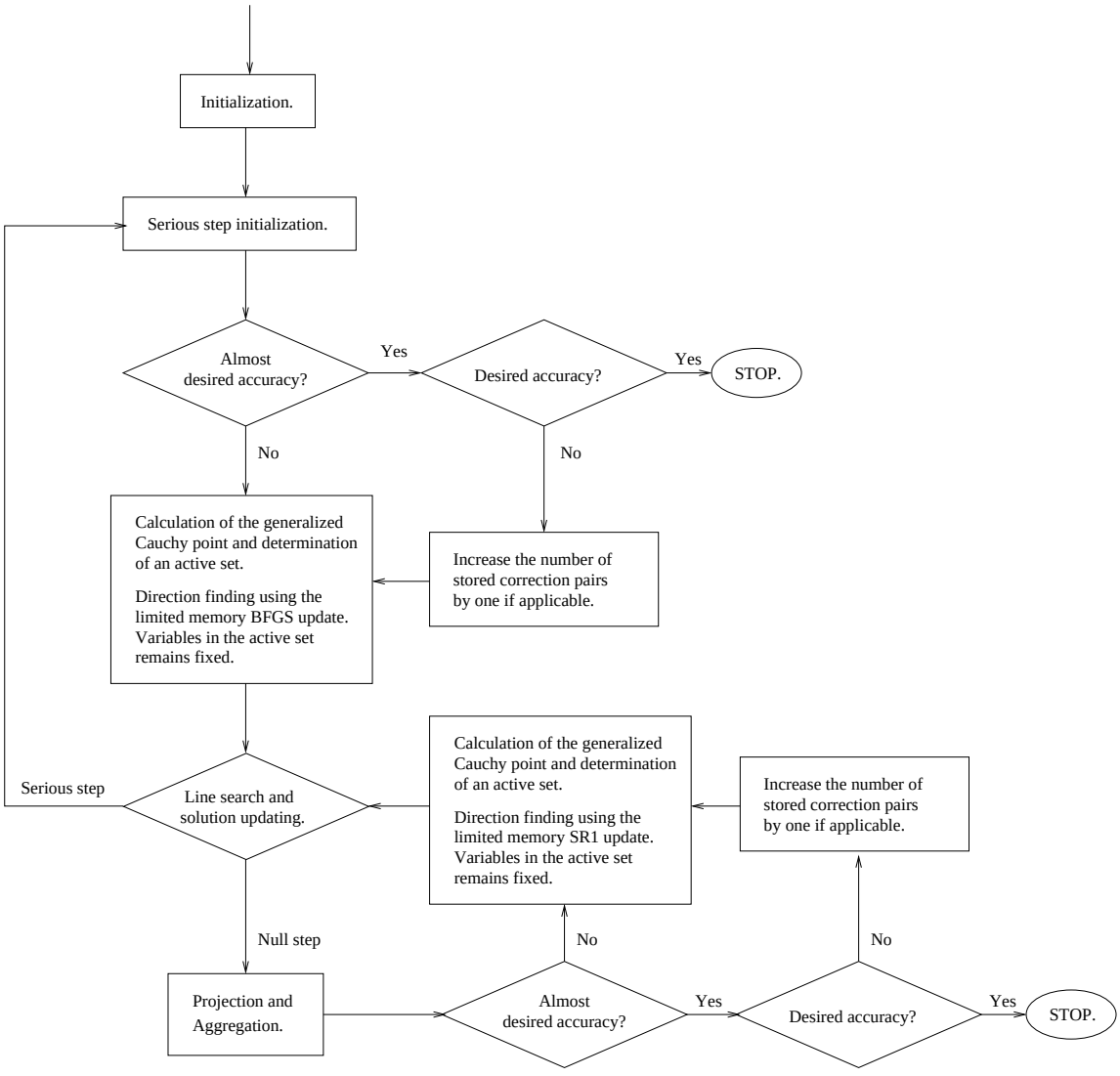

Figure 1: Adaptive limited memory bundle method with bounds.

The search direction is calculated using two-stage approach. First, we define the quadratic model function q_k that approximates the objective function at the iteration point $\boldsymbol{x}_k$ by

$$q_k(\boldsymbol{x}) = f(\boldsymbol{x}_k) + \tilde{\boldsymbol{\xi}}_k^T(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T B_k(\boldsymbol{x} - \boldsymbol{x}_k). \tag{3}$$

Here $\tilde{\boldsymbol{\xi}}_k$ is the aggregate subgradient of the objective function and B_k is a positive definite limited memory variable metric update that, in smooth case, represents the approximation of the Hessian matrix. Now, the generalized gradient projection method is used to find the generalized Cauchy point $\boldsymbol{x}_k^c$ [14] and, at the same time, to identify the active set $\mathcal{I}_A^k$ of the problem. The procedure used in LMBM-B is rather similar to that in [13]. We only use here the aggregate subgradient of the objective function instead of gradient and, in addition to the limited memory BFGS update formula, we utilize the limited memory SR1 update whenever necessary. That is, after a null step (see Figure 1).

The generalized Cauchy point at iteration k is defined as the first local minimizer (starting from $\boldsymbol{x}_k$) of the univariate piecewise quadratic function

$$\hat{q}_k(t) = q_k(\mathcal{P}_c[\boldsymbol{x}_k - t\tilde{\boldsymbol{\xi}}_k, \boldsymbol{x}^l, \boldsymbol{x}^u]), \tag{4}$$

along the projected gradient direction $\mathcal{P}_c[\boldsymbol{x}_k - t\tilde{\boldsymbol{\xi}}_k, \boldsymbol{x}^l, \boldsymbol{x}^u] - \boldsymbol{x}_k$ (see e.g. [14]). Here we have defined the projection operator $\mathcal{P}_c[\cdot]$ (component-wise) by

$$\mathcal{P}_c[\boldsymbol{x}, \boldsymbol{x}^l, \boldsymbol{x}^u]_i = \begin{cases} x_i^l, & \text{if } x_i < x_i^l \\ x_i, & \text{if } x_i \in [x_i^l, x_i^u] \\ x_i^u, & \text{if } x_i > x_i^u. \end{cases} \tag{5}$$

This operator projects the point $\boldsymbol{x}$ into the feasible region $\mathcal{F}$ defined by bounds $\boldsymbol{x}^l$ and $\boldsymbol{x}^u$. Now, if we denote by t_k^c the value of t corresponding to the first local minimum of $\hat{q}_k(t)$, the generalized Cauchy point is given by

$$\boldsymbol{x}_k^c = \mathcal{P}_c[\boldsymbol{x}_k - t_k^c\tilde{\boldsymbol{\xi}}_k, \boldsymbol{x}^l, \boldsymbol{x}^u]. \tag{6}$$

The variables whose values at $\boldsymbol{x}_k^c$ are at lower or upper bound, comprise the active set $\mathcal{I}_A^k = \{i \mid x_{k,i}^c = x_i^l \text{ or } x_{k,i}^c = x_i^u\}$. Here, we have denoted by $x_{k,i}^c$ the ith component of the vector $\boldsymbol{x}_k^c$. The calculation of this generalized Cauchy point makes it possible to add and delete several bounds from the active set during a single iteration, which may be an important feature for both nonsmooth [15] and large-scale [16] problems. For the practical computation of generalized Cauchy point see [12, 13, 17].

When the generalized Cauchy point has been found, we approximately minimize the quadratic model function (3) over the space of free variables, in other words, the variables in the active set are treated as equality constraints. The subspace minimization procedure used is in principal the same as the dual space method in [13] but, as before, we use the aggregate subgradient of the objective function and we utilize the limited memory SR1 update if the previous step taken was a null step.

We solve $\boldsymbol{d}$ from the smooth quadratic problem

$$\begin{cases} \text{minimize} & \tilde{\boldsymbol{\xi}}_k^T\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T B_k\boldsymbol{d} \\ \text{such that} & A_k^T\boldsymbol{d} = \boldsymbol{b}_k \qquad \text{and} \\ & \boldsymbol{x}^l \leq \boldsymbol{x}_k + \boldsymbol{d} \leq \boldsymbol{x}^u, \end{cases} \tag{7}$$

where A_k is the matrix of active constraints gradients at $\boldsymbol{x}_k^c$ and $\boldsymbol{b}_k = A_k^T(\boldsymbol{x}_k^c - \boldsymbol{x}_k)$. Note that A_k consists of n_A unit vectors (here n_A is the number of elements in the active set $\mathcal{I}_A^k$) and $A_k^T A_k$ is equal to identity.

We first ignore the bound constraints. The first order sufficient optimality conditions for problem (7) without bounds are

$$\tilde{\boldsymbol{\xi}}_k + B_k\boldsymbol{d}_k^* + A_k\boldsymbol{\mu}_k^* = \boldsymbol{0} \tag{8}$$

$$A_k^T\boldsymbol{d}_k^* = \boldsymbol{b}_k. \tag{9}$$

Now, by multiplying Eq.(8) by $A_k^T D_k$ (we denote by D_k the update formula that is inverse of B_k) and by using Eq.(9), we obtain

$$(A_k^T D_k A_k)\boldsymbol{\mu}_k^* = -A_k^T D_k \tilde{\boldsymbol{\xi}}_k - \boldsymbol{b}_k, \tag{10}$$

which determines Lagrange multipliers $\boldsymbol{\mu}_k^* \in \mathbb{R}^{n_A}$. The linear system (10) can be solved by utilizing the Sherman-Morrison-Woodbury formula and the compact representation of limited memory matrices (see [13]). Thus, $\boldsymbol{d}_k^*$ can be given by

$$B_k \boldsymbol{d}_k^* = -A_k \boldsymbol{\mu}_k^* - \tilde{\boldsymbol{\xi}}_k. \tag{11}$$

If there are no active variables, we simply obtain $\boldsymbol{d}_k^* = -D_k \tilde{\boldsymbol{\xi}}_k$, which is the formula used also in the original unconstrained version of the limited memory bundle method [1, 2, 3]. In the case the vector $\boldsymbol{x}_k + \boldsymbol{d}_k^*$ violates the bounds in Eq.(7), we, similarly to [13], backtrack along the line joining the infeasible point $\boldsymbol{x}_k + \boldsymbol{d}_k^*$ and the generalized Cauchy point $\boldsymbol{x}_k^c$ to regain the feasible region. That is, we first compute

$$\alpha_k^* = \min\left\{1, \max\{\alpha \mid x_i^l \leq x_{k,i}^c + \alpha(x_{k,i} + d_i^* - x_{k,i}^c) \leq x_i^u,\ i \in \mathcal{I}_F^k\}\right\}, \tag{12}$$

where $\mathcal{I}_F^k = \{j \mid j = \{1, \ldots, n\} \setminus \mathcal{I}_A^k\}$ is the set of free variables, and then we set $\bar{\boldsymbol{x}} = \boldsymbol{x}_k^c + \alpha_k^*(\boldsymbol{x}_k + \boldsymbol{d}_k^* - \boldsymbol{x}_k^c)$ and $\boldsymbol{d}_k = \bar{\boldsymbol{x}} - \boldsymbol{x}_k$. Again, see [12, 13, 17] for details of the practical computations.

LMBM-B generates a sequence of basic points $(\boldsymbol{x}_k) \subset \mathcal{F}$ together with a sequence of auxiliary points $(\boldsymbol{y}_k) \subset \mathcal{F}$. A new iteration point $\boldsymbol{x}_{k+1}$ and a new auxiliary point $\boldsymbol{y}_{k+1}$ are produced using a special line search procedure [17] such that

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k \qquad \text{and} \\ \boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + t_R^k \boldsymbol{d}_k, \qquad \text{for } k \geq 1 \end{aligned} \tag{13}$$

with $\boldsymbol{y}_1 = \boldsymbol{x}_1$, where $t_R^k \in (0, t_{max}^k]$ and $t_L^k \in [0, t_R^k]$ are step sizes, $t_{max}^k \geq 1$ is the upper bound for the step size that assures the feasibility of produced points.

A necessary condition for a serious step is to have

$$t_R^k = t_L^k > 0 \qquad \text{and} \qquad f(\boldsymbol{y}_{k+1}) \leq f(\boldsymbol{x}_k) - \varepsilon_L t_R^k w_k, \tag{14}$$

where $\varepsilon_L \in (0, 1/2)$ is a line search parameter and $w_k > 0$ represents the desirable amount of descent of f at $\boldsymbol{x}_k$. If condition (14) is satisfied, we set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ and a serious step is taken.

Otherwise, we take a null step. In this case, the usage of special line search procedure guarantees that we have

$$t_R^k > t_L^k = 0 \qquad \text{and} \qquad -\beta_{k+1} + \mathcal{P}_{\boldsymbol{x}_k}[\tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_k}[\boldsymbol{\xi}_{k+1}] \geq -\varepsilon_R w_k, \tag{15}$$

where $\varepsilon_R \in (\varepsilon_L, 1/2)$ is a line search parameter, $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$, $\mathcal{P}_{\boldsymbol{x}}[\boldsymbol{\xi}]$ denotes a stark projection of $\boldsymbol{\xi}$ at $\boldsymbol{x}$ (to be described short after), and β_{k+1} is the subgradient locality measure [18, 19] similar to bundle methods. 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 $\boldsymbol{\xi}_{k+1}$.

LMBM-B 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 used in the previous versions of LMBM [1, 2, 3, 12] is similar to that of the original variable metric bundle methods [4, 5] except that the variable metric updates are calculated using limited memory approach. However, in order to guarantee the global convergence of the bound constrained version, we need to consider projections of subgradients instead of original subgradients in the aggregation procedure. It may seem that utilization of active set $\mathcal{I}_A^k$ in the projection procedure would be a natural choice. However, active set $\mathcal{I}_A^k$ is calculated at the generalized Cauchy point $\boldsymbol{x}_k^c$ and it may change in consecutive null steps, which is highly undesirable from the view point of global convergence. Therefore, we instead calculate a very simple projection at point $\boldsymbol{x}_k$ that is not changing ($\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ in null steps). We define this projection operator $\mathcal{P}_{\boldsymbol{x}}[\cdot]$ at point $\boldsymbol{x}$ (component-wise) by

$$\mathcal{P}_{\boldsymbol{x}}[\boldsymbol{\xi}]_i = \begin{cases} 0, & \text{if } x_i^l - x_i \geq 0 \\ \xi_i, & \text{if } x_i \in (x_i^l, x_i^u) \\ 0, & \text{if } x_i^u - x_i \leq 0. \end{cases} \tag{16}$$

In what follows we call this stark projection the $\mathcal{P}_{\boldsymbol{x}}$-projection (at point $\boldsymbol{x}$).

Now, the aggregation procedure is made by determining multipliers λ_i^k satisfying $\lambda_i^k \geq 0$ for all $i \in \{1, 2, 3\}$, and $\sum_{i=1}^3 \lambda_i^k = 1$ that minimize the function

$$\varphi(\lambda_1, \lambda_2, \lambda_3) = \mathcal{P}_{\boldsymbol{x}_k}[\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k]^T D_k \mathcal{P}_{\boldsymbol{x}_k}[\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k] + 2(\lambda_2 \beta_{k+1} + \lambda_3 \tilde{\beta}_k). \tag{17}$$

Here $\boldsymbol{\xi}_m \in \partial f(\boldsymbol{x}_k)$ is the current subgradient (m denotes the index of the iteration after the latest serious step, i.e. $\boldsymbol{x}_k = \boldsymbol{x}_m$), $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$ is the auxiliary subgradient, and $\tilde{\boldsymbol{\xi}}_k$ is the current aggregate subgradient from the previous iteration ($\tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1$). In addition, β_{k+1} is the current locality measure and $\tilde{\beta}_k$ is the current aggregate locality measure ($\tilde{\beta}_1 = 0$).

The next $\tilde{\boldsymbol{\xi}}_{k+1}$ is defined as a convex combination of the subgradients mentioned above:

$$\tilde{\boldsymbol{\xi}}_{k+1} = \lambda_1^k \boldsymbol{\xi}_m + \lambda_2^k \boldsymbol{\xi}_{k+1} + \lambda_3^k \tilde{\boldsymbol{\xi}}_k \tag{18}$$

and the next $\tilde{\beta}_{k+1}$ as a convex combination of the locality measures:

$$\tilde{\beta}_{k+1} = \lambda_2^k \beta_{k+1} + \lambda_3^k \tilde{\beta}_k. \tag{19}$$

$\mathcal{P}_{\boldsymbol{x}}$-projection depends only on point $\boldsymbol{x}$ and not on the subgradient $\boldsymbol{\xi}$ on focus. Thus, $\mathcal{P}_{\boldsymbol{x}_k}[\lambda_1 \boldsymbol{\xi}_m + \lambda_2 \boldsymbol{\xi}_{k+1} + \lambda_3 \tilde{\boldsymbol{\xi}}_k] = \lambda_1 \mathcal{P}_{\boldsymbol{x}_k}[\boldsymbol{\xi}_m] + \lambda_2 \mathcal{P}_{\boldsymbol{x}_k}[\boldsymbol{\xi}_{k+1}] + \lambda_3 \mathcal{P}_{\boldsymbol{x}_k}[\tilde{\boldsymbol{\xi}}_k]$ and this makes the minimization of function (17) rather an easy task.

We utilize the limited memory approach (see e.g. [6, 7] and Appendix) in the calculation of the generalized Cauchy point, search direction, and aggregate values. The idea of limited memory matrix updating is, instead of storing the large $n \times n$ -matrices B_k and D_k, to store a certain (usually small constant) number $\hat{m}_c$ of vectors, so-called correction pairs obtained at the previous iterations of the algorithm, and to use these correction pairs to implicitly define the variable metric matrices. When the storage space available is used up, the oldest correction pair is deleted to make room for new one; thus, except for the first few iterations, we always have the $\hat{m}_c$ most recent correction pairs available.

The utilization of limited memory approach naturally means that the variable metric updates are not as accurate as if we used standard variable metric updates (see e.g. [20]). However, both the storage space required and the number of operations needed in the calculations are significantly smaller. Namely, the number of operations needed is $O(n)$ while with standard variable metric updates used in original variable metric bundle methods [4, 5], it is $O(n^2)$.

In the adaptive LMBM , first introduced in [1] and then used in [12] and here, the number of stored correction pairs $\hat{m}_c$ may change during the computation. This means that we can start the optimization with a small $\hat{m}_c$ and when we are closer to the optimal point, $\hat{m}_c$ may be increased until some predefined upper limit $\hat{m}_u$ is achieved. The aim of this adaptability is to improve the accuracy of the basic method without loosing much from efficiency, that is, without increasing computational costs too much.

The limited memory variable metric matrices used in our algorithm are represented in the compact matrix form originally described in [6]. We use both the limited memory BFGS and the limited memory SR1 update formulae in the calculations of the search direction and the aggregate values. If the previous step was a null step, the matrices D_k and B_k are formed using the limited memory SR1 updates (see Appendix, Eq.(26) and Eq.(27)). The SR1 update formulae give us a possibility to preserve the boundedness and some other properties of generated matrices that guarantee the global convergence of the method. Otherwise, since these properties are not required after a serious step, the more efficient limited memory BFGS updates (see Appendix, Eq.(21) and Eq.(24)) are employed.

The SR1-update formulae do not in general preserve the positive definiteness. Moreover, both the nonconvexity and bounds may prevent the condition $\boldsymbol{d}_i^T(\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_m) > 0$ for all $i = 1, ..., k-1$, that is the classical condition for the positive definiteness of the BFGS update, from satisfying. Thus, to maintain the positive definiteness of the generated matrices, we simply skip the individual updates (discard the newest correction pair not the oldest one) if they would violate positive definiteness.

The basic assumption for bundle methods to converge is that after a null step we have $\boldsymbol{z}^T D_{k+1} \boldsymbol{z} \leq \boldsymbol{z}^T D_k \boldsymbol{z}$ for all $\boldsymbol{z} \in \mathbb{R}^n$. In LMBM-B this is guaranteed by the special limited memory SR1 update [1, 3]. In addition, to ensure the global convergence of the method, the matrices D_k are assumed to be

uniformly positive definite and uniformly bounded (we say that a matrix is bounded if its eigenvalues lie in the compact interval that does not contain zero). This is guaranteed by adding some positive definite correction matrices to matrices D_k when necessary (i.e. we set $D_k = D_k + \rho I$ with $\rho > 0$). The detailed algorithm of globally convergent LMBM-B is given in [17].

Remark. Under mild assumptions LMBM-B is proved to be globally convergent for locally Lipschitz continuous objective functions, which are not necessarily differentiable or convex [17].

5. Numerical Experiments

We now compare the proposed globally convergent limited memory bundle method `LMBM-B` to the proximal bundle method `PBNCGC` (version 2.0, [10, 21]) in a limited number of nonsmooth large-scale test problems. We used the solver `PBNCGC` as a benchmark since the proximal bundle method is the most frequently used bundle method in nonsmooth optimization. In addition, we compared the new version `LMBM-B` to the older, non-globally convergent, version `LMBM-B-OLD` [12]. The experiments were performed in a Intel® Core™ 2 CPU 1.80GHz and all the algorithms were implemented in Fortran77 with double-precision arithmetic.

The solvers were tested with 10 nonsmooth academic minimization problems described in [22]*. The number of variables used in our experiments were 1000, 2000, and 4000. The solvers were tested with relatively small amount of stored subgradient information. That is, the size of the bundle m_ξ was set to 10 for `LMBM-B` and `LMBM-B-OLD` and to 100 for `PBNCGC` (since the previous experiments [1, 3] have shown that a larger bundle usually works better with `PBNCGC`). With both `LMBM-B` and `LMBM-B-OLD`, we used the values $\hat{m}_u = 15$ and $\hat{m}_c = 7$ due to good results of previous experiments with the older version [12]. The final accuracy parameter ε was set to 10^{-5} in all the cases and, otherwise, the default parameters of the solvers were used. In addition to the usual stopping criteria of the solvers, we terminated the experiments if the CPU time elapsed exceeded half an hour.

The results of experiments are summarized in Tables 1, 2, and 3, where Ni and Nf denote the numbers of iterations and function evaluations used, respectively, f denotes the value of the objective function at termination, and the time is an average CPU time elapsed per problem and it is given in seconds (only the accurately and successfully terminated problems were included).

In Tables 1, 2, and 3 we see the superiority of the different variants of LMBM-B when comparing the computational times; the computation times elapsed with `LMBM-B` and `LMBM-B-OLD` were usually hundreds of times shorter than those of `PBNCGC`. On the other hand, there was not a very big difference in the computational times between the different variants of LMBM-B. Although, the globally convergent version `LMBM-B` usually needed more computation time than the older one. This is due to fewer function evaluations required with `LMBM-B-OLD` (see Tables 1, 2, and 3). The increased number of function evaluations needed with `LMBM-B` is probably due to less accurate search direction caused by the stark projection of subgradients. Thus, different projection possibilities need to be studied.

The proximal bundle solver `PBNCGC` always needed less function evaluations than the different variants of LMBM-B. However, as can be seen when comparing the computational times, each individual iteration with `PBNCGC` was much more costly than that with `LMBM-B` or `LMBM-B-OLD`. Indeed, with `PBNCGC` all the problems with 4000 variables but two (problems 2 and 6) were terminated because the time limit exceeded. This also explains the inaccurate results obtained with `PBNCGC` (see Table 3).

The new variant `LMBM-B` failed to solve two of the problems (problems 1 and 2) with any number of variables tested. These failures were quite predictable, since both of these problems are reported to be difficult to solve with limited memory bundle method even without the bound constraints [2].

To sum up, the new solver `LMBM-B` did not beat up `LMBM-B-OLD` due to larger number of function evaluations needed. Although in theory, the new version is globally convergent and the older version is not, there was no significant improvement in the accuracy or robustness of the method. However, when comparing to `PBNCGC`, the new solver `LMBM-B` was substantially faster.

*All the problems can be downloaded from the website http://napsu.karmitsa.fi/lmbm

Table 1: Results for bound constrained problems with 1000 variables.

Solver	LMBM-B		LMBM-B-OLD		PBNCGC	
Problem	Ni/Nf	f	Ni/Nf	f	Ni/Nf	f
1	-/-	fail	11346/11385	0.01	-/-	fail
2	-/-	fail	676/825	0.26651	18/19	$8.2 \cdot 10^{-6}$
3	286/2029	−1396.08	35/62	−1395.45	18/23	−1396.12
4	157/921	2334.75	64/86	2334.75	25/26	2334.75
5	95/632	2042.62	63/252	2042.63	25/26	2042.62
6	278/291	0.09547	526/526	0.09531	9/25	0.40523
7	293/3106	99.9000	160/694	99.9001	300/395	99.9192
8	120/648	−698.121	74/310	−698.099	35/36	−698.172
9	129/895	8.45406	55/123	8.45415	46/74	8.45407
10	142/812	147.301	63/137	147.299	22/33	147.299
Time	1.07		0.21*		67.48	

* Problem 1 that took 71.91 sec to compute is not included in average CPU time of LMBM-B-OLD.

Table 2: Results for bound constrained problems with 2000 variables.

Solver	LMBM-B		LMBM-B-OLD		PBNCGC	
Problem	Ni/Nf	f	Ni/Nf	f	Ni/Nf	f
1	-/-	fail	-/-	fail	-/-	fail
2	-/-	fail	3375/3453	0.38118	20/22	$3.0 \cdot 10^{-6}$
3	112/644	−2793.50	36/71	−2791.41	19/24	−2793.63
4	150/749	4671.97	74/139	4671.97	22/23	4671.97
5	101/421	4087.25	74/276	4087.26	28/29	4087.25
6	517/522	0.09531	-/-	fail	-/-	fail
7	63/427	200.717	53/91	199.979	69/82	200.251
8	138/686	−1396.05	68/164	−1396.68	31/32	−1396.93
9	86/500	16.9159	63/177	16.9068	35/55	16.9065
10	146/812	294.911	108/110	294.792	36/48	294.793
Time	1.11		0.68		540.14	

Table 3: Results for bound constrained problems with 4000 variables.

Solver	LMBM-B		LMBM-B-OLD		PBNCGC	
Problem	Ni/Nf	f	Ni/Nf	f	Ni/Nf	f
1	-/-	fail	-/-	fail	-/-	fail
2	-/-	fail	-/-	fail	24/28	$1.8 \cdot 10^{-6}$
3	66/403	−5555.69	42/71	−5587.79	15/20	−5588.65
4	172/946	9346.40	88/126	9346.40	13/14	9346.54
5	149/686	8176.57	100/152	8176.57	13/14	8176.63
6	-/-	fail	-/-	fail	19/38	28.5408
7	281/2369	399.900	-/-	fail	14/17	404.086
8	120/521	−2794.24	123/329	−2794.39	11/12	−2784.70
9	129/639	33.8113	76/128	33.8270	13/17	35.6548
10	204/1117	589.780	95/230	589.946	13/17	593.702
Time	5.57		1.90		1562.28	

6. Conclusions

In this paper, we have described a new variant LMBM-B of the limited memory bundle method for bound constrained nonsmooth large-scale optimization. The preliminary numerical experiments confirm that LMBM-B is efficient for both convex and nonconvex large-scale nonsmooth optimization problems. With large numbers of variables it used significantly less CPU time than the proximal bundle method tested. Moreover, the difference between the computational times of this globally convergent variant and the previous (non convergent) version of the method was not as large as we foreboded.

A drawback of the new variant is clearly the increased number of function evaluations needed. This is probably due to less accurate search direction caused by the stark projection of subgradients. Thus, different projection possibilities need to be studied.

The fact that LMBM-B only generates feasible points may be essential in the case the objective function or the subgradient values are undefined or difficult to compute if some of the constraints are violated. Furthermore, it can be an advantage in many industrial applications, where function evaluation may be very expensive. Since any intermediate solution can be employed, the iterations can be stopped whenever the result is satisfactory. Due to this feasibility, the efficiency of the method, and the fact that the objective function need not to be differentiable or convex, we expect LMBM-B to be very useful in solving optimization problems arising in real world modeling.

Acknowledgements

I would like to thank Prof. Marko M. Mäkelä for his valuable comments. This work was financially supported by University of Turku (Finland).

Appendix

The limited memory variable metric matrices used in our algorithm are represented in the compact matrix form originally described in [6].

Let us denote by $\hat{m}_c$ the user-specified maximum number of stored correction pairs ($3 \leq \hat{m}_c$) and by $\hat{m}_k = \min\{k-1, \hat{m}_c\}$ the current number of stored correction pairs. Then the $n \times \hat{m}_k$ dimensional correction matrices S_k and U_k are defined by

$$S_k = \begin{bmatrix} \boldsymbol{s}_{k-\hat{m}_k} & \dots & \boldsymbol{s}_{k-1} \end{bmatrix} \qquad \text{and} \qquad U_k = \begin{bmatrix} \boldsymbol{u}_{k-\hat{m}_k} & \dots & \boldsymbol{u}_{k-1} \end{bmatrix}, \tag{20}$$

where the correction pairs $\boldsymbol{s}_i = \boldsymbol{y}_{i+1} - \boldsymbol{x}_i$ and $\boldsymbol{u}_i = \boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_m$, ($i < k$ and m denotes the index ot the iteration after the latest serious step) are obtained at the previous iterations.

The inverse limited memory BFGS update is defined by the formula

$$D_k = \vartheta_k I + \begin{bmatrix} S_k & \vartheta_k U_k \end{bmatrix} \begin{bmatrix} (R_k^{-1})^T (C_k + \vartheta_k U_k^T U_k) R_k^{-1} & -(R_k^{-1})^T \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^T \\ \vartheta_k U_k^T \end{bmatrix}, \tag{21}$$

where R_k is an upper triangular matrix of order $\hat{m}_k$ given by the form

$$(R_k)_{ij} = \begin{cases} (\boldsymbol{s}_{k-\hat{m}_k-1+i})^T (\boldsymbol{u}_{k-\hat{m}_k-1+j}), & \text{if } i \leq j \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

C_k is a diagonal matrix of order $\hat{m}_k$ such that

$$C_k = \operatorname{diag}\left[\boldsymbol{s}_{k-\hat{m}_k}^T \boldsymbol{u}_{k-\hat{m}_k}, \dots, \boldsymbol{s}_{k-1}^T \boldsymbol{u}_{k-1}\right], \tag{23}$$

and ϑ_k is a positive scaling parameter.

The similar representation for the direct limited memory BFGS update can be written by

$$B_k = \frac{1}{\vartheta_k} I - \begin{bmatrix} \frac{1}{\vartheta_k} S_k & U_k \end{bmatrix} \begin{bmatrix} \frac{1}{\vartheta} S_k^T S_k & L_k \\ L_k^T & -C_k \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{\vartheta_k} S_k^T \\ U_k^T \end{bmatrix}, \tag{24}$$

where

$$L_k = S_k^T U_k - R_k. \tag{25}$$

In addition, the inverse limited memory SR1 update is defined by

$$D_k = \vartheta_k I - (\vartheta_k U_k - S_k)(\vartheta_k U_k^T U_k - R_k - R_k^T + C_k)^{-1}(\vartheta_k U_k - S_k)^T. \tag{26}$$

and, correspondingly, the direct SR1 update is defined by

$$B_k = \frac{1}{\vartheta_k} I + (U_k - \frac{1}{\vartheta_k} S_k)(L_k + L_k^T + C_k - \frac{1}{\vartheta_k} S_k^T S_k)^{-1}(U_k - \frac{1}{\vartheta_k} S_k)^T \tag{27}$$

With SR1 updates we use the value $\vartheta_k = 1$ for all k.

In our proposal, the individual updates that would violate positive definiteness are skipped.

7. References

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

[2] M. Haarala, K. Miettinen, and M. M. Mäkelä, New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods and Software*, 2004, 19(6), 673–692.

[3] N. Haarala, K. Miettinen, and M. M. Mäkelä. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 2007, 109(1), 181–205.

[4] L. Lukšan and J. Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 1999, 102(3), 593–613.

[5] J. Vlček and L. Lukšan. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications*, 2001, 111(2), 407–430.

[6] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 1994, 63, 129–156.

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

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

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

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

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

[12] M. Haarala and M. M. Mäkelä. Limited memory bundle algorithm for large bound constrained nonsmooth minimization problems. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, B. 1/2006 University of Jyväskylä, Jyväskylä, 2006.

[13] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 1995, 16(5), 1190–1208.

[14] A. R. Conn, N. I. M. Gould, and P. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 1988, 25(2), 433–460.

[15] E. R. Panier. An active set method for solving linearly constrained nonsmooth optimization problems. *Mathematical Programming*, 1987, 37, 269–292.

[16] A. R. Conn, N. I. M. Gould, and P. L. Toint. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, 1988, 50(182), 399–430.

[17] N. Karmitsa and M. M. Mäkelä. Globally convergent limited memory bundle algorithm for nondifferentiable programming subject to box constraints. TUCS Technical Report, No. 882, Turku Centre for Computer Science, Turku, 2008.

[18] C. Lemaréchal, J.-J. Strodiot, and A. Bihain. On a bundle algorithm for nonsmooth optimization. In O. L. Mangasarian, R. R. Mayer, and S. M. Robinson, editors, *Nonlinear Programming*, pages 285–281. Academic Press, New York, 1981.

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

[20] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, Chichester, 2nd edition, 1987.

[21] M. M. Mäkelä. Multiobjective proximal bundle method for nonconvex nonsmooth optimization: Fortran subroutine MPBNGC 2.0. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, B. 13/2003 University of Jyväskylä, Jyväskylä, 2003.

[22] N. Karmitsa. Test problems for large-scale nonsmooth minimization. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, B. 4/2007 University of Jyväskylä, Jyväskylä, 2007.