Limited Memory Bundle Algorithm for Inequality Constrained Nondifferentiable Optimization

Napsu Karmitsa       Marko M. Mäkelä

Montaz M. Ali

University of Jyväskylä
Department of Mathematical Information Technology
P.O. Box 35 (Agora)
FI–40014 University of Jyväskylä
FINLAND
fax +358 14 260 2731
http://www.mit.jyu.fi/
Limited Memory Bundle Algorithm for Inequality Constrained Nondifferentiable Optimization

Napsu Karmitsa† Marko M. Mäkelä‡ Montaz M. Ali§

Abstract

Many practical optimization problems involve nonsmooth (that is, not necessarily differentiable) functions of hundreds or thousands of variables with various constraints. In this paper, we describe a new efficient adaptive limited memory interior point bundle method for large, possible nonconvex, nonsmooth inequality constrained optimization. The method is a hybrid of the nonsmooth variable metric bundle method and the smooth limited memory variable metric method, and the constraint handling is based on the primal-dual feasible direction interior point approach. The preliminary numerical experiments to be presented confirm the effectiveness of the method.

Keywords: Nonsmooth optimization, large-scale problems, constrained optimization, bundle methods, limited memory methods, feasible direction interior point methods.

1 Introduction

In this paper, we propose a new adaptive limited memory interior point bundle algorithm for solving large inequality constrained nonsmooth optimization problems. We write this problem as

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g_i(x) \leq 0, \quad \text{for } i = 1, \ldots, p,
\end{align*}
\]

where the objective function \( f : \mathbb{R}^n \to \mathbb{R} \) and the constraint functions \( g_i : \mathbb{R}^n \to \mathbb{R} \) for each \( i \in \mathcal{P} = \{1, \ldots, p\} \) are supposed to be locally Lipschitz continuous and

---

*The work was financially supported by University of the Witwatersrand, South Africa; University of Jyväskylä, Finland; and University of Turku, Finland.
†Department of Mathematical Information Technology, PO Box 35 (Agora), FI-40014 University of Jyväskylä, Finland, hamasi@mit.jyu.fi
‡Department of Mathematics, University of Turku, FI-20014 Turku, Finland, makela@utu.fi
§School of Computational and Applied Mathematics, University of the Witwatersrand, Private Bag 3, Wits 2050, Johannesburg, South Africa, mali@cam.wits.ac.za
the number of variables \( n \) is supposed to be large. Note that no differentiability or convexity assumptions are made.

Many practical optimization problems involve nonsmooth functions with large amounts of variables (see, e.g., [2, 3, 30]). These kind of problems are in general difficult to solve even when they are unconstrained. The direct application of smooth gradient-based methods to nonsmooth problems usually leads to a failure in convergence, in optimality conditions, or in gradient approximation (see, e.g., [6, 24]). On the other hand, derivative free methods like genetic algorithms (see, e.g., [10]) or Powell’s method (see, e.g., [7]) may be unreliable and become inefficient whenever the dimension of the problem increases. Thus, special tools for solving large-scale nonsmooth optimization problems are needed.

At the moment, different variants of bundle methods (see, e.g., [20, 31, 36]) are regarded as the most effective and reliable globally convergent methods for nonsmooth optimization. The basic idea of these methods is to approximate the subdifferential [6] of the nonsmooth objective by gathering subgradients (generalized gradients) from previous iterations into a bundle. A descent search direction can then be found as a solution of a quadratic subproblem and the global convergence of the methods, with limited number of stored subgradients, can be guaranteed by using a subgradient aggregation strategy [20], which accumulates information from previous iterations.

Naturally, the presence of constraints makes nonsmooth optimization problems even more complex. The problems with simple constraints (such as bound or linear constraints) may be solved by including the constraints directly to the quadratic subproblem (see, e.g., [21, 22]) and the more generally constrained problems may be solved by applying bundle methods to an equivalent unconstrained problem with an exact penalty objective (see, e.g., [19, 23]) or by using bundle methods with so-called improvement function (see, e.g., [20, 25, 31, 35]) or with filter [8, 18]. However, all these methods are suitable only for relatively small problems and most of them (excluding [21, 31]) are capable of handling only convex problems.

In [11, 13, 14] we have proposed a limited memory bundle method for general, possibly nonconvex, nonsmooth large-scale unconstrained optimization. The method is a hybrid of the variable metric bundle methods [27, 38] and the limited memory variable metric methods (see, e.g., [5, 9, 33]), 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. In [12] the variant of the method suitable for bound constrained problems was introduced.

In this paper, we combine the adaptive limited memory bundle method [11] with a modification of the feasible direction interior point method by Herskovits and Santos [15, 16] in order to make the method suitable for solving more generally constrained problems. We have used the approach of the feasible direction interior point method because it does not involve penalty or barrier functions, active set strategies, or quadratic subproblems but merely a solution of two internal linear systems with the same matrix at each iteration. By this way, the computational demand of the method is kept relatively low and we obtain a method suitable for
solving large-scale problems.

Our method is feasible, in that, given an initial point that satisfy the constraints, it construct a sequence of points which all satisfy them as well. This kind of interior-point approach is essential in case the objective function and/or the constraint functions are not defined in infeasible points. 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.

The rest of this paper is organized as follows. In the following section, we first recall some basic results and definitions of nonsmooth analysis. In Section 3, we describe the adaptive limited memory interior point bundle method for inequality constrained optimization and, in Section 4, we analyze its convergence properties. In Section 5, some preliminary results of numerical experiments are presented and finally, in Section 6, we conclude. A detailed description of limited memory matrix updating is given in Appendix.

## 2 Background and Optimality Conditions

In this section we give the optimality conditions for nonsmooth inequality constrained optimization problem (1). Thereby, we first recall some basic definitions and results from nonsmooth analysis based on Clarke [6]. For details and proofs we refer to [6, 31].

In what follows, we assume that all the functions considered are locally Lipschitz continuous and by Rademacher’s Theorem a locally Lipschitz continuous function is differentiable almost everywhere. Thus, we can use the following theorem to define the subdifferential.

**Theorem 2.1.** Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) be locally Lipschitz continuous at \( x \in \mathbb{R}^n \). Then the subdifferential of \( f \) at \( x \) is a set \( \partial f(x) \) of vectors \( \xi_f \in \mathbb{R}^n \) such that

\[
\partial f(x) = \text{conv} \{ \xi_f \in \mathbb{R}^n \mid \text{there exists } (x_i) \subset \mathbb{R}^n \setminus \Omega_f \text{ such that } x_i \to x \text{ and } \nabla f(x_i) \to \xi_f \}.
\]

Here “conv” denotes the convex hull of the set and \( \Omega_f \) is a set where \( f \) fails to be differentiable. Each vector \( \xi_f \in \partial f(x) \) is called a subgradient of \( f \) at \( x \).

The subdifferential \( \partial f(x) \) is a nonempty, convex, and compact set such that \( \partial f(x) \subset B(0; L) \), where \( L > 0 \) is the Lipschitz constant of \( f \) at \( x \).

In iterative optimization methods it is necessary to find a direction such that the objective function values decrease when moving in that direction. Next we define a descent direction.

**Definition 2.2.** The direction \( d \in \mathbb{R}^n \) is said to be a descent direction for \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) at \( x \in \mathbb{R}^n \), if there exists \( \varepsilon > 0 \) such that for all \( t \in (0, \varepsilon] \)

\[
f(x + td) < f(x).
\]
Lemma 2.3. Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) be a locally Lipschitz continuous function at \( x \in \mathbb{R}^n \). The direction \( d \in \mathbb{R}^n \) is a descent direction for \( f \) at \( x \) if \( \xi^T f d < 0 \) for all \( \xi_f \in \partial f(x) \).

In constrained optimization it is not enough to find any descent direction, since we are not allowed to violate the constraints. Thus, we need to define the feasible direction.

Definition 2.4. The direction \( d \in \mathbb{R}^n \) is said to be a feasible direction for problem (1), if there exists \( \varepsilon > 0 \) such that for all \( t \in (0, \varepsilon] \)

\[
x + td \in S,
\]

where \( S = \{ x \in \mathbb{R}^n \mid g_i(x) \leq 0 \text{ for all } i \in P \} \) is the feasible region for problem (1).

Lemma 2.5. Let \( g_i : \mathbb{R}^n \rightarrow \mathbb{R} \) for each \( i \in P \) be locally Lipschitz continuous functions at \( x \in S \). The direction \( d \in \mathbb{R}^n \) is a feasible direction for problem (1) if \( \xi^T g_i d < 0 \) for all \( \xi_{g_i} \in \partial g_i(x) \) such that \( g_i(x) = 0 \).

Proof. Follows directly from the fact that by Lemma 2.3 direction \( d \) is a descent direction for all \( g_i \) such that \( g_i(x) = 0 \). \( \square \)

In order to formulate necessary Karush-Kuhn-Tucker (KKT) type optimality conditions, we first need to define the following regularity assumption.

Definition 2.6. Problem (1) satisfies the Cottle constraint qualification at \( x \), if either \( g_i(x) < 0 \) for all \( i \in P \) or \( 0 \notin \text{conv}\{ \partial g_i(x) \mid g_i(x) = 0 \} \).

Theorem 2.7. (KKT optimality condition.) Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( g_i : \mathbb{R}^n \rightarrow \mathbb{R} \) for each \( i \in P \) be locally Lipschitz continuous functions at \( x \in \mathbb{R}^n \) and suppose that problem (1) satisfies the Cottle constraint qualification. If \( x \) is a local minimum of (1), then there exist Lagrange multipliers \( \mu_i \geq 0 \) such that \( \mu_i g_i(x) = 0 \) for all \( i \in P \) and

\[
0 \in \partial f(x) + \sum_{i \in P} \mu_i \partial g_i(x).
\]

A point \( x \) is said to be a KKT point associated to problem (1) if it is feasible and it satisfies the KKT optimality condition (Theorem 2.7). If all the functions \( f \) and \( g_i \) \( (i \in P) \) are convex, the KKT optimality condition is sufficient and the KKT point is a global minimum for problem (1).

In what follows we denote by \( L(x, \mu) = f(x) + \sum_{i \in P} \mu_i g_i(x) \) the Lagrangian of problem (1) and by \( \partial L(x, \mu) \) its subdifferential. Moreover, we denote by \( \partial L(x, \mu) = \partial f(x) + \sum_{i \in P} \mu_i \partial g_i(x) \). Note that we have the following inclusion \( \partial L(x, \mu) \subset \partial L(x, \mu) \) (see, e.g., [31]).

3 Method

In this section, we describe the adaptive limited memory interior point bundle method for inequality constrained large-scale nonsmooth optimization. We start by giving a simple flowchart (in Figure 1) to point out the basic ideas of the algorithm.
The limited memory interior point bundle method is characterized by the usage of null steps together with the aggregation of subgradients. At each iteration we first solve a linear system of KKT type optimality conditions in order to find a descent direction for the objective. After that, we (by means of computing a feasible direction) perturb the linear system to deflect the descent direction to a feasible descent direction. Finally, the line search is performed along this direction either to obtain a new interior point with lower objective or to take a null step. Using null steps gives sufficient information about the nonsmooth objective in the case the search direction is not “good enough”. On the other hand, a simple aggregation of subgradients guarantees the convergence of the aggregate subgradients to zero and makes it possible to evaluate a termination criterion.

The limited memory approach (see, e.g., [5, 9, 33]) is utilized both in the calculation of the search direction and the aggregate values. The idea of limited memory matrix updating is that instead of storing the large matrices we store a certain (usually small constant) number of vectors, so-called correction pairs, obtained at the previous iterations of the algorithm and we use these vectors to implicitly de-
fine the variable metric matrices. When the storage space available is used up, 
the oldest correction pairs are deleted to make room for new ones. In the adap-
tive limited memory bundle method [11] the number of stored correction pairs may 
change during the computation. This means that we can start the optimization with 
a small number of stored correction pairs and then, when we are closer to the opti-
mal point, the number of stored correction pairs may be increased until some upper 
limit 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 computa-
tional costs too much.

3.1 Limited memory interior point bundle method.

We now describe with more details the limited memory interior point bundle al-
gorithm for solving nonsmooth optimization problems of type (1). The algorithm 
to be presented generates a sequence of basic points \((x_k) \subseteq \text{int } S\) together with a 
sequence of auxiliary points \((y_k) \subseteq \text{int } S\). A new iteration point \(x_{k+1}\) and a new 
 auxiliary point \(y_{k+1}\) are produced using a special constrained line search procedure 
such that

\[
\begin{align*}
x_{k+1} &= x_k + t_k^L d_k \quad \text{and} \\
y_{k+1} &= x_k + t_k^R d_k, 
\end{align*}
\]

for \(k \geq 1\) with \(y_1 = x_1 \in \text{int } S\), where \(t_k^R \in (0, t_{\text{max}}]\) and \(t_k^L \in [0, t_k^R]\) are step sizes, \(t_{\text{max}} \geq 1\) is 
the upper bound for the step size, and \(d_k\) is a search direction.

A necessary condition for a serious step is to have \(t_k^R = t_k^L > 0\),

\[
\begin{align*}
f(y_{k+1}) &\leq f(x_k) - \varepsilon L t_k^L w_k^1, \quad \text{and} \\
g_i(y_{k+1}) &< 0 
\end{align*}
\]

for all \(i \in P\), where \(\varepsilon L \in (0, 1/2)\) is a linesearch parameter and \(w_k^1 > 0\) represents the desirable 
amount of descent of \(f\) at \(x_k\). If condition (3) is satisfied, we have \(x_{k+1} = y_{k+1}\) and a 
serious step is taken.

Otherwise, we take a null step. In this case, the usage of special line search pro-
cedure guarantees that we have \(t_k^R = t_k^L = 0\),

\[
\begin{align*}
-\beta_{L,k+1} - \tilde{\xi}_{L,k}^T D_k \xi_{L,k+1} &\geq -\varepsilon R w_k^1, \quad \text{and} \\
g_i(y_{k+1}) &< 0 
\end{align*}
\]

for all \(i \in P\).

Here \(\varepsilon R \in (\varepsilon L, 1/2)\) is a line search parameter, \(\mu_k\) is a dual variable, \(\tilde{\xi}_{L,k} \in \partial L(x_k, \mu_k)\) 
is the aggregate subgradient of the Lagrangian, \(\xi_{L,k+1} \in \partial L(y_{k+1}, \mu_k)\) is the new 
 auxiliary subgradient of the Lagrangian, \(D_k\) is a positive definite limited memory 
variable metric update that, in smooth case, approximates the inverse of the Hessian 
of the Lagrangian associated to problem (1), and \(\beta_{L,k+1}\) is the subgradient locality 
measure [26, 32] defined by

\[
\beta_{L,k+1} = \max \{ |L(x_k, \mu_k) - L(y_{k+1}, \mu_k) + (y_{k+1} - x_k)^T \xi_{L,k+1}|, \gamma \|y_{k+1} - x_k\|^2 \}.
\]
In the case of a null step, we have \( x_{k+1} = x_k \) but information about the objective function is increased because we store the auxiliary point \( y_{k+1} \) and the corresponding auxiliary subgradient \( \xi_{L,k+1} \).

**Search direction.** The search direction \( d_k \) is calculated using a modification of the feasible direction interior point method [15, 16]. First, we formulate the necessary KKT type optimality conditions for problem (1) by

\[
\begin{align*}
\tilde{\xi}_f + \tilde{\xi}_g \mu &= 0, \quad (6) \\
G(x) \mu &= 0, \quad (7) \\
g(x) &\leq 0, \quad (8) \\
\mu &\geq 0, \quad (9)
\end{align*}
\]

where \( \tilde{\xi}_f \) is the aggregate subgradient of the objective function, \( \tilde{\xi}_g \) is the \( p \times n \) matrix of the aggregate subgradients of the constraint functions, \( g(x) \) denotes the \( p \)-vector of constraints values, and \( G(x) \) denotes a diagonal \( p \times p \)-matrix such that \( G(x) = \text{diag}[g_1(x), \ldots, g_p(x)] \). Moreover, \( \mu \) denotes the \( p \)-vector of dual variables.

A quasi-Newton’s -type iteration to solve the system of equations (6), (7) can be defined by

\[
\begin{bmatrix}
B_k \xi_{g,k} \\
\Lambda_k \xi_{g,k} G(x_k)
\end{bmatrix}
\begin{bmatrix}
\alpha_k x_{k+1} - x_k \\
\alpha_k \mu_{k+1} - \mu_k
\end{bmatrix}
= -\begin{bmatrix}
\tilde{\xi}_{f,k} + \tilde{\xi}_{g,k} \mu_k \\
G(x_k) \mu_k
\end{bmatrix},
\]

where \( (x_k, \mu_k) \) is the starting point of the iteration, \( (x_{k+1}^\alpha, \mu_{k+1}^\alpha) \) is the new estimate (index \( \alpha \) refers to descent direction calculations), \( \Lambda_k \) denotes the diagonal \( p \times p \)-matrix such that \( \Lambda_k = \text{diag}[\mu_{1,k}, \ldots, \mu_{p,k}] \), and \( B_k \) is a positive definite limited memory variable metric update \( (B_k = D_k^{-1}) \). We solve the system of equations (6), (7) such that (8), (9) are verified at each iteration.

The iterations in (10) are modified such that, for the given interior pair \( (x_k, \mu_k) \), we obtain a new interior estimate with a better objective. For this purpose, a primal direction \( d_k^\alpha = x_{k+1}^\alpha - x_k \) is defined. Now, we can write (10) as a linear system in \( d_k^\alpha \) and \( \mu_{k+1}^\alpha \) by

\[
\begin{align*}
B_k d_k^\alpha + \xi_{g,k} \mu_{k+1}^\alpha &= -\tilde{\xi}_{f,k} \quad (11) \\
\Lambda_k \xi_{g,k} d_k^\alpha + G(x_k) \mu_{k+1}^\alpha &= 0 \quad (12)
\end{align*}
\]

and we have the following result for \( d_k^\alpha \):

**Lemma 3.1.** Suppose that \( \mu_k > 0 \) for all \( k \geq 1 \). The direction \( d_k^\alpha \) defined by (11) and (12) satisfies

\[
(d_k^\alpha)^T \tilde{\xi}_{f,k} \leq -(d_k^\alpha)^T B_k d_k^\alpha \quad \text{for all } k \geq 1.
\]

**Proof.** By multiplying both sides of (11) by \( d_k^\alpha \), we obtain

\[
(d_k^\alpha)^T \tilde{\xi}_{f,k} = -(d_k^\alpha)^T B_k d_k^\alpha - (d_k^\alpha)^T \xi_{g,k} \mu_{k+1}^\alpha.
\]
Now, by using (12), this can be written
\[
(d_k^\alpha)^T \tilde{\xi}_{f,k} = -(d_k^\alpha)^T B_k d_k^\alpha + (\mu_{k+1}^\alpha)^T \Lambda_k^{-1} G(x_k) \mu_{k+1}^\alpha
\]
and the result follows from the fact that \(\Lambda_k^{-1} G(x_k)\) is negative definite for all \(k\) due to strict feasibility of \(x_k\). \(\square\)

Due to the preceding lemma, the positiveness of \(\mu_{k+1}\) and the positive definiteness of \(B_k\) used in our proposal, we have
\[
\tilde{\xi}_{f,k}^T d_k^\alpha \leq 0 \quad \text{for all } k \geq 1.
\]

Thus, by Lemma 2.3 and the fact that \(\tilde{\xi}_{f,k}\) is a convex combination of the previous subgradients (the proof is similar to that of Lemma 3.2 in [38]) direction \(d_k^\alpha\) seems to be a suitable choice for a descent direction. Nevertheless, from (12) it comes out that when any constraint goes to zero \(d_k^\alpha\) tends to be orthogonal to the aggregate subgradient of that constraint and we may obtain an infeasible direction. Thus, we deflect \(d_k^\alpha\) towards the interior of the feasible region by means of the vector \(d_k^\beta\) defined by the linear system
\[
\begin{align*}
B_k d_k^\beta + \tilde{\xi}_{g,k}^\beta \mu_{k+1}^\beta &= 0 \\
\Lambda_k \tilde{\xi}_{g,k}^\beta d_k^\beta + G(x_k) \mu_{k+1}^\beta &= -\mu_k.
\end{align*}
\]

Now, the search direction can be calculated by
\[
d_k = d_k^\alpha + \rho_k d_k^\beta.
\]

Here the deflection bound \(\rho_k > 0\) (see (17) and (18)) is selected such that the condition
\[
\tilde{\xi}_{f,k}^T d_k \leq \nu \tilde{\xi}_{f,k}^T d_k^\alpha
\]
with \(\nu \in (0, 1)\) is satisfied (see Lemma 4.3 in [15]). Thus, we have \(\tilde{\xi}_{f,k}^T d_k \leq 0\) and, from (12), (14), and (15), we obtain \(\tilde{\xi}_{g,k}^T d_k = -\rho_k < 0\) for all active constraints (i.e., for constraints with \(g_i(x_k) = 0, i \in P\)). Therefore, due to Lemmas 2.3 and 2.5 the direction \(d_k\) seems to be a suitable choice for a feasible descent direction.

There are different possibilities of updating \(\mu_k\) (see, e.g., [15] or [1] in a slightly different contents). In this work we have adopted and modified the combination of two used in [15], since these updating rules guarantee the boundedness of \(\mu_k\) (see [15]). More specifically, we initialize \(\mu_{i,1}\) for all \(i \in P\) by \(\mu_{i,1} = \min\{-1/g_i(x_1), \mu_{\text{max}}\}\), where \(\mu_{\text{max}} > 0\) is a predefined upper limit. Note that we assume the starting point \(x_1\) to be strictly feasible and, thus, \(g_i(x_1) < 0\) for all \(i \in P\). In the subsequent iterations we set \(\mu_{i,k+1} = \max\{\mu_{i,k+1}^\alpha, \epsilon ||d_k^\beta||^2\}\) with some \(\epsilon > 0\), if the next step is a serious step, and \(\mu_{i,k+1} = \mu_{i,k}\) if the next step is a null step. If \(g_i(x) \geq g_{\text{max}}\) and \(\mu_{i,k+1} < \mu_{\text{min}}\) for some predefined \(\mu_{\text{min}} > 0\) and \(g_{\text{max}} < 0\), we set \(\mu_{i,k+1} = \mu_{\text{min}}\).
The limited memory interior point bundle algorithm uses, for the direction determination, the original subgradients after the serious step and the aggregate subgradients after the null step. The aggregation procedure (see Step 5 in Algorithm 3.1) is similar to that of the original variable metric bundle methods [27, 38] except two matters. First, we use the subgradients and the variable metric approximations of the Lagrangian instead of those of the objective function and, secondly, the variable metric updates are calculated using limited memory approach, namely the limited memory BFGS and SR1 updates: If the previous step was a null step, the matrix $D_k$ (i.e., $B_k^{-1}$) is formed using the limited memory SR1 update (see Appendix, (30)), since this update formula gives us a possibility to preserve the boundedness and some other properties of generated matrices that are needed to guarantee the convergence of aggregate subgradients to zero. Otherwise, since these properties are not required after a serious step, the more efficient limited memory BFGS update (see Appendix, (29)) is employed. The individual updates that would violate positive definiteness are skipped (for more details, see [11, 13, 14] and Appendix).

**Algorithm.** We now present a model algorithm for the adaptive limited memory interior point bundle method for solving the inequality constrained minimization problems of type (1). In what follows, we assume that at every point $x \in \mathbb{R}^n$ we can evaluate the values $f(x)$ and $g_i(x)$ for all $i \in P$ and the corresponding arbitrary subgradients $\xi_f \in \partial f(x)$, $\xi_{g_i} \in \partial g_i(x)$, and $\xi_L \in \partial L(x, \mu)$ (for some $\mu \in \mathbb{R}^p$). In addition, we assume that the problem considered satisfies Cottle constraint qualification (Def. 2.6) and that the feasible region $S \subset \mathbb{R}^n$ is nonempty and has an interior.

**ALGORITHM 3.1.** *(Limited memory interior point bundle method.)*

**Data:** Choose the final accuracy tolerances $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$, the positive line search parameters $\varepsilon_L \in (0, 1/2)$ and $\varepsilon_R \in (\varepsilon_L, 1/2)$, and the distance measure parameter $\gamma \geq 0$ (with $\gamma = 0$ if all the functions involved are convex). Select the lower and the upper bounds $t_{min} \in (0, 1)$ and $t_{max} > 1$ for serious steps. Select the control parameters $C > 0$ for the length of the direction vector and $\nu > 0$ and $\nu \in (0, 1)$ for the deflection bound. Select the parameter $\epsilon > 0$, the limit $g_{\text{max}} < 0$, and the auxiliary lower and upper bounds $\mu_{\text{min}} > 0$ and $\mu_{\text{max}} > 0$ for dual variables. Select an upper limit $\hat{m}_u \geq 3$ for the number of stored correction pairs.

**Step 0:** *(Initialization.)* Choose a strictly feasible starting point $x_1 \in \text{int } S$ and a positive initial vector $\mu_1 \in \mathbb{R}^p$ (e.g., $\mu_{i,1} = \min \{-1/g_i(x_1), \mu_{\text{max}}\}$). Choose an initial maximum number of stored correction pairs $\hat{m}_c$ (for serious steps). Select the iteration counter $k = 1$. Set $y_1 = x_1$, $\mu_{i,1} = \mu_1$, and $\beta_{L,1} = 0$. Compute $f_1 = f(x_1)$, $\xi_{f,1} \in \partial f(x_1)$, $g_1 = g(x_1)$, $\xi_{g,1} \in \partial g(x_1)$ (i.e., a generalized Jacobian [6]), and $\xi_{L,1} \in \partial L(x_1, \mu_1)$. Set the iteration counter $k = 1$. 


Step 1: (Serious step initialization.) Set the aggregate subgradients $\tilde{\xi}_{f,k} = \xi_{f,k}$, $\tilde{\xi}_{g,k} = \xi_{g,k}$, and $\tilde{\xi}_{L,k} = \xi_{L,k}$, and the aggregate subgradient locality measure $\tilde{\beta}_{L,k} = 0$. Set an index for the serious step $m = k$. 

Step 2: (Stopping criterion.) Calculate $w^1_k$ and $w^2_k$ by

\[
   w^1_k = \xi_{L,k}^T D_k \tilde{\xi}_{L,k} + 2 \tilde{\beta}_{L,k} \\
   w^2_k = -\sum_{i \in P} \mu_{i,k} g_{i,k},
\]

respectively. Use the limited memory BFGS update for calculation of $D_k$ if $m = k$ and the limited memory SR1 update, otherwise (see Appendix). If $w^1_k \leq \varepsilon_1$ and $w^2_k \leq \varepsilon_2$, then stop with $x_k$ as the final solution. Otherwise, if $w^1_k \leq 10^3 \varepsilon_1$ and $\hat{m}_c < \hat{m}_u$, set $\hat{m}_c = \hat{m}_c + 1$. 

Step 3: (Direction finding.)

(i) (Descent direction.) Solve the values $d^\alpha_k \in \mathbb{R}^n$ and $\mu_{k+1}^\alpha \in \mathbb{R}^p$ satisfying the linear equations (11) and (12) using the same updating formula for $D_k$ (i.e., for $B_k^{-1}$) as in step 2. If 

\[
   \|d^\alpha_k\| \leq \varepsilon_1,
\]

then stop with $x_k$ as the final solution.

(ii) (Feasible direction.) Solve the values $d^\beta_k \in \mathbb{R}^n$ and $\mu_{k+1}^\beta \in \mathbb{R}^p$ satisfying the linear equations (13) and (14) (use the same updating formula as before). If $\tilde{\xi}_{f,k} d^\beta_k > 0$, set 

\[
   \rho_k = \min \left\{ \varrho \|d^\alpha_k\|^2, \frac{(\nu - 1) \tilde{\xi}_{f,k} d^\alpha_k}{\tilde{\xi}_{f,k} d^\beta_k} \right\}. 
\]

Otherwise, set 

\[
   \rho_k = \varrho \|d^\alpha_k\|^2. 
\]

(iii) (Feasible descent direction.) Compute the search direction

\[
   d_k = d^\alpha_k + \rho_k d^\beta_k. 
\]

Step 4: (Line search and solution updating.) Set the scaling parameter for the length of the direction vector and for line search $\theta_k = \min \{ 1, C/\|d_k\| \}$. Choose the initial step size $t^I_k \in [t_{\min}, t_{\max})$. Determine the step sizes $t^I_k \in (0, t^I_k)$ and $t^I_k \in \ldots$
by the constrained line search Algorithm 3.2. Set the corresponding values

\[ x_{k+1} = x_k + t^k \theta_k d_k, \quad f_{k+1} = f(x_{k+1}), \quad g_{k+1} = g(x_{k+1}), \]
\[ y_{k+1} = x_k + t^k \theta_k d_k, \quad \xi_{f,k+1} \in \partial f(y_{k+1}), \quad \xi_{g,k+1} \in \partial g(y_{k+1}), \]

and

\[ \hat{\xi}_{L,k+1} \in \partial L(y_{k+1}, \mu_k). \]

Set \( u_k = \hat{\xi}_{L,k+1} - \xi_{L,m} \) and \( s_k = y_{k+1} - x_k = t^k \theta_k d_k \) and update the limited memory matrices \( U_{k+1} \) and \( S_{k+1} \) (see Appendix).

If condition (3) is valid (i.e., we take a serious step), then set

\[ \mu_{i,k+1} = \max \{ \mu^\alpha_{i,k+1}, \epsilon ||d^\alpha_k||^2 \} \quad \text{for all } i \in \mathcal{P} \]

and, if \( g_i(y_{k+1}) \geq g_{\max} \) and \( \mu_{i,k+1} < \mu_{\min} \) for some \( i \in \mathcal{P} \), then set \( \mu_{i,k+1} = \mu_{\min} \). Set \( \beta_{L,k+1} = 0, k = k + 1 \), and go to Step 1. Otherwise (i.e., condition (4) is valid), set \( \xi_{L,k+1} = \hat{\xi}_{L,k+1} \), calculate the locality measure \( \beta_{L,k+1} \) by (5), and set \( \mu_{k+1} = \mu_k \).

**Step 5:** (Aggregation.) Determine 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

\[ \varphi(\lambda_1, \lambda_2, \lambda_3) = (\lambda_1 \xi_{L,m} + \lambda_2 \xi_{L,k+1} + \lambda_3 \tilde{\xi}_{L,k})^T D_k (\lambda_1 \xi_{L,m} + \lambda_2 \xi_{L,k+1} + \lambda_3 \tilde{\xi}_{L,k}) + 2(\lambda_2 \beta_{L,k+1} + \lambda_3 \tilde{\beta}_{L,k}), \]

where, as before, \( D_k \) is calculated by the same updating formula as in Step 2. Set

\[ \tilde{\xi}_{f,k+1} = \lambda_1^k \xi_{f,m} + \lambda_2^k \xi_{f,k+1} + \lambda_3^k \tilde{\xi}_{f,k}, \]
\[ \tilde{\xi}_{g_i,k+1} = \lambda_1^k \xi_{g_i,m} + \lambda_2^k \xi_{g_i,k+1} + \lambda_3^k \tilde{\xi}_{g_i,k} \quad \text{for all } i \in \mathcal{P}, \]
\[ \tilde{\xi}_{L,k+1} = \lambda_1^k \xi_{L,m} + \lambda_2^k \xi_{L,k+1} + \lambda_3^k \tilde{\xi}_{L,k}, \]

and

\[ \tilde{\beta}_{L,k+1} = \lambda_2^k \beta_{L,k+1} + \lambda_3^k \tilde{\beta}_{L,k}. \]

Set \( k = k + 1 \) and go to Step 2.
Assumptions to matrices. To ensure the convergence of aggregate subgradients to zero, we assume that matrices $D_k$ are 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 requires some modifications to the model algorithm, for instance, corrections of matrices $D_k$ when necessary. In this way we obtain more complicated algorithm which, in unconstrained case, is described in detail in [11, 14]. The basic assumption for bundle method to converge, that is, after a null step we have $z^T D_{k+1} z \leq z^T D_k z$ for all $z \in \mathbb{R}^n$, is guaranteed by the special limited memory SR1 update [11, 14]. In addition to these, we assume that there exist positive numbers $\omega_1$ and $\omega_2$ such that $\omega_1 \|d\|^2 \leq d^T B_k d \leq \omega_2 \|d\|^2$ for all $d \in \mathbb{R}^n$ (see [37] for less restrictive conditions for $B_k$).

Line search procedure. The initial step size $t^I_k \in [t_{\min}, t_{\max})$ (see Step 4 in Algorithm 3.1) is selected by using a bundle containing auxiliary points and corresponding function values and subgradients. The procedure used is similar to that in the original variable metric bundle method for nonconvex objective functions [38]. We now present the line search algorithm, which is used to determine the step sizes $t^L_k$ and $t^R_k$ in the limited memory interior point bundle method. The line search procedure used is rather similar to that given in [11, 14] which, on the other hand, was derived from [38]. However, in interior point algorithms we are not allowed to violate constraints. Thus, we need to extend the line search procedure used in the previous variants of the limited memory bundle method to the constrained case.

**Algorithm 3.2.** (Modified line search for inequality constrained problems).

**Data:** Suppose that we have the current iteration point $x_k$, the current dual variables $\mu_k$, the current search direction $d_k$, the current scaling parameter $\theta_k \in (0, 1]$, the current vector $\hat{\xi}_{L,k}^T D_k$, and the positive line search parameters $\varepsilon_L \in (0, 1/2)$, $\varepsilon_R \in (\varepsilon_L, 1/2)$, $\varepsilon_A \in (0, \varepsilon_R - \varepsilon_L)$, and $\varepsilon_T \in (\varepsilon_L, \varepsilon_R - \varepsilon_A)$ available. In addition, suppose that we have given the initial step size $t^I_k$, an auxiliary lower bound for serious steps $t_{\min} \in (0, 1)$, the distance measure parameter $\gamma \geq 0$, the desirable amount of descent $w^1_k$, the maximum number of additional interpolations $i_{\max}$, and the number of consecutive null steps $i_{null} \geq 0$.

**Step 0:** (Initialization.) Set $t_A = 0$, $t = t_U = t^I_k$, and $i_I = 0$, and calculate the interpolation parameter

$$\kappa = 1 - \frac{1}{2(1 - \varepsilon_T)}.$$

**Step 1:** (New values.) Compute $f(x_k + t\theta_k d_k)$, $g_i(x_k + t\theta_k d_k)$ for all $i \in P$, $\hat{\xi}_L \in \partial L(x_k + t\theta_k d_k, \mu_k)$, and

$$\beta_L = \max \left\{ |L(x_k, \mu_k) - L(x_k + t\theta_k d_k, \mu_k) + t\theta_k d_k^T \hat{\xi}_L|, \gamma (t\theta_k \|d_k\|^2) \right\}. \quad (22)$$
If
\[ g_i(x_k + t\theta_k d_k) \geq 0 \quad \text{for some } i \in \mathcal{P}, \]
then set \( t_U = t \) and go to Step 5. Else, if \( f(x_k + t\theta_k d_k) \leq f(x_k) - \varepsilon_T t w^1_k \), then set \( t_A = t \). Otherwise, set \( t_U = t \).

Step 2: (Serious step.) If
\[ f(x_k + t\theta_k d_k) \leq f(x_k) - \varepsilon_L t w^1_k, \]
and either
\[ t \geq t_{\min} \quad \text{or} \quad \beta_L > \varepsilon_A w^1_k, \]
then set \( t_R = t_L^k = t \) and stop.

Step 3: (Test for additional interpolation.) If \( f(x_k + t\theta_k d_k) > f(x_k), i_{\text{null}} > 0 \), and \( i_I < i_{\max} \), then set \( i_I = i_I + 1 \) and go to Step 5.

Step 4: (Null step.) If
\[ -\beta_L - \xi^T_{L,k} D_k \hat{\xi}_L \geq -\varepsilon_R w^1_k, \]
then set \( t_R = t, t_L^k = 0 \) and stop.

Step 5: (Interpolation.) If \( t_A = 0 \), then set
\[ t = \max \left\{ t_U, \frac{-\frac{1}{2}t_U^2 w^1_k}{f(x_k) - f(x_k + t\theta_k d_k) - t_U w^1_k} \right\}. \]
Otherwise, set \( t = \frac{1}{2}(t_A + t_U) \). Go to Step 1.

It can be proved that Algorithm 3.2 terminates in a finite number of iterations (the proof is rather similar to that given in [38]) if the problem satisfies the following modified semi-smoothness assumption: For all \( \xi_L \in \partial L(x, \mu_k) \) and for any \( x \in \mathbb{R}^n \) and \( d \in \mathbb{R}^n \), and sequences \( (\hat{\xi}_{L,i}) \subset \mathbb{R}^n \) and \( (t_i) \subset \mathbb{R}_+ \) satisfying \( \hat{\xi}_{L,i} \in \partial L(x + t_i d, \mu_k) \) and \( t_i \downarrow 0 \), we have
\[ -\limsup_{i \to \infty} \xi^T_{L,k} D_k \hat{\xi}_{L,i} \geq \liminf_{i \to \infty} \frac{f(x + t_i d) - f(x)}{t_i}, \]
where \( D \) is the inverse variable metric approximation calculated at point \( x \). Note that if we have no constraints present (i.e., \( p = 0 \) in problem (1)), then this modified semi-smoothness assumption reverts very similar to classical semi-smoothness assumption [4].

On the output of Algorithm 3.2 (see Steps 2 and 4), the step sizes \( t^k_L \) and \( t^k_R \) satisfy the serious descent criterion
\[ f(x_{k+1}) - f(x_k) \leq -\varepsilon_L t^k_L w^1_k \]
and, in the case of \( t^k_L = 0 \) (a null step), also condition (4). Moreover, in both cases, for all \( i \in \mathcal{P} \), we have \( g_i(y_{k+1}) < 0 \).
4 Convergence Analysis

In this section, we study the convergence properties of Algorithm 3.1. In addition to assuming that all the functions involved are locally Lipschitz continuous, the set $S \cap \{ x \in \mathbb{R}^n \mid f(x) \leq f(x_1) \}$ is supposed to be compact and the problem is assumed to satisfy Cottle constraint qualification (Def. 2.6). Furthermore, we assume that each execution of the line search procedure is finite (i.e., the modified semismoothness assumption is valid) and that the matrices $D_k$ and $B_k$ satisfy the assumptions given before.

We start the theoretical analysis of limited memory interior point bundle method by noting that the solutions of linear systems (11), (12), and (13), (14) are unique. After that, we study the case when the algorithm terminates after a finite number of iterations: we prove that if Algorithm 3.1 stops at iteration $k$, then the point $x_k$ is a KKT point for problem (1) (note that if all the functions involved are convex, this is also a global minimum for the problem). Finally, we prove that every accumulation point $\bar{x}$ of the sequence $(x_k) \subset S$ is a KKT point for problem (1) provided the complementary condition (see Theorem 2.7) is satisfied. For these purposes, we assume that the final accuracy tolerances $\varepsilon_1$ and $\varepsilon_2$ are equal to zero.

Remark 4.1. The sequence $(x_k)$ is bounded by assumption and the monotonicity of the sequence $(f_k)$ obtained due to serious descent criterion (23). Since $x_{k+1} = y_{k+1}$ for serious steps and $\|y_{k+1} - x_{k+1}\| \leq t_{\text{max}} C$ for null steps by (2) and due to the fact that we use the scaled direction vector $\theta_k d_k$ with $\theta_k = \min \{ 1, C/\|d_k\| \}$ and predefined $C > 0$ in the line search, the sequence $(y_k)$ is also bounded. By the local boundedness and the upper semicontinuity of subdifferential we obtain the boundedness of subgradients $\xi_{L,k}, \xi_{f,k}$, and $\xi_{g_i,k}$ for all $i \in P$, as well as all their convex combinations (see [6]).

The fact that the solutions of linear systems (11), (12), and (13), (14) are unique is a consequence of Lemma 3.1 in [34] stated as follows (using the notation of this paper)

Lemma 4.1. For any vector $x \in S$, any positive definite matrix $B \in \mathbb{R}^{n \times n}$ and any non-negative vector $\mu \in \mathbb{R}^p$ such that $\mu_i > 0$ if $g_i(x) = 0$, the matrix

$$
M(x, B, \mu) = \begin{bmatrix}
B & \hat{\xi}_g \\
\Lambda \xi_g^T & G(x)
\end{bmatrix}
$$

is nonsingular.

Remark 4.2. Since the matrix $B_k$ is bounded by assumption, $x_k$ lies in a compact set, and $\mu_k$ is bounded (see [15]), the matrix $M(x_k, B_k, \mu_k)$ is bounded away from zero and, thus, $d_k^\alpha, \mu_k^\alpha, d_k^\beta$, and $\mu_k^\beta$ are bounded from above.

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

$$(\tilde{\xi}_{L,k}, \tilde{\alpha}_k) = \sum_{j=1}^{k} \lambda^{k,j} (\xi_{L,j}, \|y_j - x_k\|), \quad \sum_{j=1}^{k} \lambda^{k,j} = 1, \quad \text{and} \quad \tilde{\beta}_{L,k} \geq \gamma \tilde{\alpha}_k^2. $$

**PROOF.** See the proof of Lemma 3.2 in [38].

**LEMMA 4.3.** Let $\bar{x} \in \mathbb{R}^n$ be given and suppose that there exist a function $\psi : \mathbb{R}^n \to \mathbb{R}$, vectors $\tilde{\zeta}, \tilde{\xi}_j, \tilde{y}_j$, and numbers $\tilde{\lambda}_j \geq 0$ for $j = 1, \ldots, l$, $l \geq 1$, such that

$$(\tilde{\zeta}, 0) = \sum_{j=1}^{l} \tilde{\lambda}_j (\tilde{\xi}_j, \|\tilde{y}_j - \bar{x}\|),$$

$$\tilde{\xi}_j \in \partial \psi(\tilde{y}_j), \quad j = 1, \ldots, l, \quad \text{and} \quad \sum_{j=1}^{l} \tilde{\lambda}_j = 1.$$ 

Then $\tilde{\zeta} \in \partial \psi(\bar{x})$.

**PROOF.** See the proof of Lemma 3.3 in [38].

**THEOREM 4.4.** If Algorithm 3.1 terminates at the $k$th iteration, then the point $x_k$ is a KKT point for problem (1).

**PROOF.** If Algorithm 3.1 terminates at Step 3(i), then the fact $\varepsilon_1 = 0$ implies that $d_k^a$ is zero. Due to (11), (12), and strict feasibility of $x_k$, we have $\tilde{\xi}_{L,k} + \tilde{\xi}_{g,k} \mu_{k+1} = 0$ with $\mu_{k+1} = 0$ and, thus, the point $x_k$ is a KKT point for problem (1).

Let us now assume that Algorithm 3.1 terminates at Step 2. We point out first that $\tilde{\beta}_{L,k} \geq 0$ for all $k$ by (5), (21), and Step 1 in Algorithm 3.1. Due to (16), the positive definiteness of $D_k$, and the correction term $\sigma I$ with $\sigma \in (0, 1/2)$ that is added to matrix $D_k$, if necessary (see [11, 14]), we have

$$w_k^1 \geq 2 \tilde{\beta}_{L,k} \quad \text{and} \quad w_k^1 \geq \sigma \|\tilde{\xi}_{L,k}\|^2. \quad (24)$$

Since Algorithm 3.1 terminates at Step 2, we have $w_k^1 = 0$. Thus, $\tilde{\xi}_{L,k} = 0$ and $\tilde{\beta}_{L,k} = \tilde{\alpha}_k = 0$ by (24) and Lemma 4.2. Now, by Lemma 4.2 and by using Lemma 4.3 with

$$\bar{x} = x_k, \quad l = k, \quad \tilde{\zeta} = \tilde{\xi}_{L,k},$$

$$\tilde{\xi}_j = \xi_{L,j}, \quad \tilde{y}_j = y_j, \quad \tilde{\lambda}_j = \lambda^{k,j} \quad \text{for} \quad j \leq k,$$

and

$$\partial \psi(\tilde{y}_j) = \partial L(y_j, \mu_j)$$

we obtain $0 = \tilde{\xi}_{L,k} \in \partial L(x_k, \mu_k) \subset \partial \tilde{L}(x_k, \mu_k)$.

Moreover, since Algorithm 3.1 stops at Step 2, we have $w_k = 0$. By feasibility of $x_k$ and positiveness of $\mu_k$ it follows that $G(x_k) \mu_k = 0$ and, thus, $x_k$ is a KKT point for problem (1).
From now on, we suppose that Algorithm 3.1 does not terminate, that is, $d_k^a \neq 0$ and $w_k^1 > 0$ for all $k$.

**Lemma 4.5.** Suppose that there exist a point $\bar{x} \in S$, a vector $\bar{\mu} \in \mathbb{R}^p$ and an infinite set $K \subset \{1, 2, \ldots\}$ such that $(x_k)_{k \in K} \to \bar{x}$, $(\mu_k)_{k \in K} \to \bar{\mu}$, and $(w_k^1)_{k \in K} \to 0$, then $0 \in \partial L(\bar{x}, \bar{\mu})$. Moreover, if $w_k^2 \to 0$, then $\bar{x}$ is a KKT point for problem (1).

**Proof.** By using the subdifferential of Lagrangian instead of objective function the first part of the proof is similar to the proof of Lemma 3.4 in [38]. Due to feasibility of $\bar{x}$ and positiveness of $\bar{\mu}$ the second part follows directly from the definition of KKT points (see Theorem 2.7). \qed

**Lemma 4.6.** Suppose that the number of serious steps is finite and the last serious step occurred at the iteration $m - 1$. Then there exists a number $k^* \geq m$, such that

$$w_{k+1}^1 = \xi_{L,k+1}^T D_{k+1} \xi_{L,k+1} \leq \xi_{L,k+1}^T D_k \xi_{L,k+1} + 2 \beta_{L,k+1}$$

and

$$\text{tr}(D_k) < \frac{3}{2} n$$

for all $k \geq k^*$, where $\text{tr}(D_k)$ denotes the trace of matrix $D_k$.

**Proof.** The result is due to safeguarded SR1 update used (see [11, 14]). See the proof of Lemma 7 in [14]. \qed

**Lemma 4.7.** Suppose that the number of serious steps is finite and the last serious step occurred at the iteration $m - 1$. Then, $0 \in \partial L(x_m, \mu_m)$. Moreover, if $w_k^2 \to 0$, then $x_m$ is a KKT point for problem (1).

**Proof.** From (16), (19), (20), (21), and Lemma 4.6 we obtain

$$w_{k+1}^1 = \xi_{L,k+1}^T D_{k+1} \xi_{L,k+1} + 2 \beta_{L,k+1} \leq \xi_{L,k+1}^T D_k \xi_{L,k+1} + 2 \beta_{L,k+1} \leq \xi_{L,k}^T D_k \xi_{L,k} + 2 \beta_{L,k} = w_k^1$$

for $k \geq k^*$ with $k^*$ defined in Lemma 4.6. The last inequality in (27) follows from the fact that the pair $(\xi_{L,k+1}, \beta_{L,k+1})$ minimizes function (19) over all convex combinations of pairs $(\xi_{L,m}, \beta_{L,m})$, $(\xi_{L,k+1}, \beta_{L,k+1})$, and $(\xi_{L,k}, \beta_{L,k})$. In addition, the line search procedure guarantees that we have

$$- \beta_{L,k+1} - \xi_{L,k}^T D_k \xi_{L,k+1} \geq - R w_k^1$$

for all $k \geq m$. Now, due to boundedness of $\xi_{L,k+1}, \xi_{L,k}$, and $D_k$ (see Remark 4.1 and Lemma 4.6) it can be proved that $w_k^1 \to 0$ (the proof is rather similar to the proof, part(ii), of Lemma 3.6 in [38]). Now, since $x_k = x_m$ and $\mu_k = \mu_m$ for all $k \geq m$, we have $x_k \to x_m$ and $\mu_k \to \mu_m$. Therefore, by Lemma 4.5 we have $0 \in \partial L(x_m, \mu_m)$ and $x_m$ is a KKT point for problem (1), if $w_k^2 \to 0$. \qed
THEOREM 4.8. For any accumulation point \( \bar{x} \) of the sequence \( (x_k) \) we have \( 0 \in \partial L(\bar{x}, \bar{\mu}) \), where \( \bar{\mu} \) is an accumulation point of the sequence \( (\mu_k) \). Moreover, if \( w_k^2 \to 0 \), then \( \bar{x} \) is a KKT point for problem (1).

PROOF. Let \( \bar{x} \) and \( \bar{\mu} \) be accumulation points of \( (x_k) \) and \( (\mu_k) \), respectively. Further, let \( K \subset \{1, 2, \ldots \} \) be an infinite set such that \( (x_k)_{k \in K} \to \bar{x} \) and \( (\mu_k)_{k \in K} \to \bar{\mu} \). Note that due to boundedness of \( \mu_k \) (see [15]) such a set exists. In view of Lemma 4.7, we can restrict our consideration to the case where the number of serious steps (with \( t_L^k > 0 \)) is infinite. Let us denote

\[
K' = \{ k \mid t_L^k > 0, \text{ there exists } i \in K, k \leq i \text{ such that } x_i = x_k \text{ and } \mu_i = \mu_k \}.
\]

Obviously, \( K' \) is infinite, \( (x_k)_{k \in K'} \to \bar{x} \), and \( (\mu_k)_{k \in K'} \to \bar{\mu} \). The continuity of \( f \) implies that \( (f_k)_{k \in K'} \to f(\bar{x}) \) and, thus, \( f_k \downarrow f(\bar{x}) \) by the monotonicity of the sequence \( (f_k) \) obtained due to serious descent criterion (23). Using the fact that \( t_L^k \geq 0 \) for all \( k \geq 1 \) and condition (23), we obtain

\[
0 \leq \varepsilon_L t_L^k w_k^1 \leq f_k - f_{k+1} \to 0 \quad \text{for } k \geq 1. \tag{28}
\]

If the set \( K_1 = \{ k \in K' \mid t_L^k \geq t_{\min} \} \) is infinite, then \( (w_k^1)_{k \in K_1} \to 0 \) by (28) and \( (x_k)_{k \in K_1} \to \bar{x} \) and \( (\mu_k)_{k \in K_1} \to \bar{\mu} \). Thus, by Lemma 4.5 we have \( 0 \in \partial L(\bar{x}, \bar{\mu}) \) and \( \bar{x} \) is a KKT point for problem (1) if \( w_k^2 \to 0 \).

If the set \( K_1 \) is finite, then the set \( K_2 = \{ k \in K' \mid \beta_{L,k+1} > \varepsilon_A w_k^1 \} \) has to be infinite (see Algorithm 3.2, Step 2). To the contrary, let us assume that

\[
w_k^1 \geq \delta > 0, \quad \text{for all } k \in K_2.
\]

From (28), we have \( (t_L^k)_{k \in K_2} \to 0 \) and Step 4 in Algorithm 3.1 implies

\[
\|x_{k+1} - x_k\| = t_L^k \theta_k \|d_k\| \leq t_L^k C
\]

for all \( k \geq 1 \). Thus, we have \( (\|x_{k+1} - x_k\|)_{k \in K_2} \to 0 \). By (22), (28), the boundedness of \( \xi_{L,k} \) and \( \mu_k^0 \) (see Remarks 4.1 and 4.2), and the fact that all the constraint functions are locally Lipschitz continuous (i.e., we have \( g_i(x_k) - g_i(x_{k+1}) \leq L \|x_k - x_{k+1}\| \) for all \( i \in P \) with some Lipschitz constant \( L > 0 \)), we obtain \( (\beta_{L,k+1})_{k \in K_2} \to 0 \), which is in contradiction with

\[
\varepsilon_A \delta \leq \varepsilon_A w_k^1 < \beta_{L,k+1}, \quad k \in K_2.
\]

Therefore, there exists an infinite set \( K_3 \subset K_2 \) such that \( (w_k^1)_{k \in K_3} \to 0 \), \( (x_k)_{k \in K_3} \to \bar{x} \), and \( (\mu_k)_{k \in K_3} \to \bar{\mu} \). By Lemma 4.5, we have \( 0 \in \partial L(\bar{x}, \bar{\mu}) \) and \( \bar{x} \) is a KKT point for problem (1), if \( w_k^2 \to 0 \). \( \square \)

5 Numerical Experiments

In this section we compare the limited memory interior point bundle method \( \text{LMBM-IP} \) to the proximal bundle method \( \text{PBNCGC} \) with improvement function [31]. 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 compare LMBM–IP to the bound constrained version of the method LMBM–B [12]. The experiments were performed in a Intel® Pentium® 4 CPU 3.20GHz and all the algorithms were implemented in Fortran77 with double-precision arithmetic.

The number of variables used in all the experiments was 1000, and the solvers were tested with relatively small sizes of the bundles \( m_\xi \), that is, \( m_\xi = 2 \) for LMBM–IP and LMBM–B and \( m_\xi = 100 \) for PBNCGC (since the previous experiments [11, 14] have shown that a larger bundle usually works better with PBNCGC). For the limited memory bundle solvers LMBM–IP and LMBM–B the upper limit for the stored correction pairs \( \hat{m}_u \) was set to 15 and the initial maximum number of stored corrections pairs \( \hat{m}_c \) was set to 7. The (first) final accuracy tolerance \( \varepsilon_1 = 10^{-5} \) was used in all the cases. In addition to the usual stopping criteria of the solvers, we terminated the experiments if the CPU time elapsed exceeded half an hour.

Through the numerical experiments the following parameters were used with the solver LMBM–IP:

\[
\rho = 10^{-9}, \quad \nu = 0.99, \quad \epsilon = 10^{-12}, \quad g_{\text{max}} = 0.001, \quad \text{and} \quad \varepsilon_2 = 10^{-4}.
\]

In addition, the parameters \( \mu_{\text{min}} \) and \( \mu_{\text{max}} \) were selected from the values \( \mu_{\text{max}} = 2.0, 10.0, \) and 1000.0 and \( \mu_{\text{min}} = 0.001, \) and 0.1. With all the solvers the distance measure parameter value \( \gamma = 0 \) was used with convex problems and the value \( \gamma = 0.5 \) with nonconvex problems. Otherwise, the default values of the parameters were used.

**Bound constrained problems.** The solvers were first tested with 10 nonsmooth academic minimization problems described in [13]. Half of these problems (problems 1–5) were convex and the other half (problems 6–10) were nonconvex. The problems in [13] are unconstrained but we inclosed the additional bounds

\[
x_i^* + 0.1 \leq x_i \leq x_i^* + 1.1 \quad \text{for all odd } i \text{ such that } i \leq 100,
\]

where \( x^* \) denotes the solution for the unconstrained problem. If the original starting point given in [13] was not feasible, we simply projected it to the feasible region and, since the starting point for LMBM–IP have to be strictly feasible, added an additional safeguard of 0.0001.

The results of the bound constrained experiments are given in Table 1, where \( \text{Ni} \) and \( \text{Nf} \) 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).

To sum up, the new solver LMBM–IP do not beat up LMBM–B that has specially been designed for solving large-scale nonsmooth bound constrained problems. However, even if there exists a special way of dealing with bound constraints in PBNCGC, the solver LMBM–IP was on the average 36 times faster than it.
### Table 1: Results for bound constrained problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solver</th>
<th>Ni/Nf</th>
<th>f</th>
<th>Ni/Nf</th>
<th>f</th>
<th>Ni/Nf</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-/-</td>
<td>fail</td>
<td>17364/17652</td>
<td>0.01000</td>
<td>8647/8671</td>
<td>0.01000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>70/791</td>
<td>0.05148</td>
<td>143/690</td>
<td>0.05187</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>153/1956</td>
<td>0.05148</td>
<td>143/690</td>
<td>0.05187</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>114/1391</td>
<td>0.05148</td>
<td>143/690</td>
<td>0.05187</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>66/929</td>
<td>0.05148</td>
<td>143/690</td>
<td>0.05187</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>54/552</td>
<td>0.09531</td>
<td>143/690</td>
<td>0.09531</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>108/1563</td>
<td>0.09531</td>
<td>143/690</td>
<td>0.09531</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>259/2505</td>
<td>-705.456</td>
<td>143/690</td>
<td>-705.456</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>32/456</td>
<td>0.09531</td>
<td>143/690</td>
<td>0.09531</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>130/1227</td>
<td>0.09531</td>
<td>143/690</td>
<td>0.09531</td>
<td>19/26</td>
<td>0.00006</td>
<td></td>
</tr>
</tbody>
</table>

#### Inequality constrained problems.

In addition to bound constraint problems we tested the solvers LMBM-IP and PBNCGC with 50 inequality constrained problems. The problems were constructed by combining the problems in [13] to the constraints given in Appendix. The constraints were selected such that the original unconstrained minima of problems are not feasible. Note that, due to nonconvexity of the constraints, all the inequality constrained problems used in our experiments were nonconvex. This means that there may exist more than one local minimum for these problems.

The results of the inequality constrained experiments are given in Table 2. The number of the problem in Table 2 is constructed by first taking the number of the problem in [13] and then the number of the constraint in Appendix. Note that due to large dimension, nonconvexity, nonsmoothness, and nonlinear/nonsmooth inequality constraints these problems are very difficult to solve.

In Table 2 we used the term “fail” in three different failure terminations: first, if there was some numerical difficulties that prevent solvers from working (“fail-1”, in problem 7-2); second, if, by practically speaking, no optimization occurred (“fail-2”, in almost all problems with constraints 3 and 4 with PBNCGC and in problems 1-3 and 1-5 with LMBM-IP); and, third, if the solution found was not even near the optimal solution (“fail-3”, in the rest of the failed problems in Table 2). In this last case, the most common reason for termination with PBNCGC was that the time was up and with LMBM-IP it was that the value of the objective function is not changed enough in last ten iterations.

At the end of Table 2 we have calculated the average numbers of iterations and function calls as well as the average CPU time elapsed per problem. In each cases, we omitted the problem that used most function evaluations since, for example, with LMBM-IP problem 9-3 used 91% of all function evaluations required.

In Table 2 we can see the superiority of LMBM-IP when comparing the computational times; the computational time elapsed with LMBM-IP was an average about 40 times shorter than that of PBNCGC. It also succeed to solve many more problems (80%) than PBNCGC (50%). However, there were some inaccuracy results obtained.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Solver</th>
<th>Ni/Nf</th>
<th>$f$</th>
<th>Time</th>
<th>Ni/Nf</th>
<th>$f$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>LMBM-1P</td>
<td>4914/10791</td>
<td>0.500065</td>
<td>14.25</td>
<td>-/-</td>
<td>-/-</td>
<td>fail-3</td>
</tr>
<tr>
<td>2-1</td>
<td>PBNCGC</td>
<td>110/829</td>
<td>0.043072</td>
<td>12.49</td>
<td>2807/9012</td>
<td>0.000163</td>
<td>202.60</td>
</tr>
<tr>
<td>3-1</td>
<td></td>
<td>244/2721</td>
<td>-/- fail-3</td>
<td>1.51</td>
<td>7673/22478</td>
<td>-/- fail-3</td>
<td>1800.21</td>
</tr>
<tr>
<td>4-1</td>
<td></td>
<td>745/9015</td>
<td>2003.24</td>
<td>6.11</td>
<td>-/-</td>
<td>fail-3</td>
<td></td>
</tr>
<tr>
<td>5-1</td>
<td></td>
<td>230/1853</td>
<td>1998.36</td>
<td>1.35</td>
<td>1882/4199</td>
<td>1998.80</td>
<td>137.61</td>
</tr>
<tr>
<td>6-1</td>
<td></td>
<td>137/1060</td>
<td>0.534851</td>
<td>0.61</td>
<td>68/123</td>
<td>0.535092</td>
<td>0.73</td>
</tr>
<tr>
<td>7-1</td>
<td></td>
<td>351/3728</td>
<td>5.00248</td>
<td>7.78</td>
<td>-/-</td>
<td>fail-3</td>
<td></td>
</tr>
<tr>
<td>8-1</td>
<td></td>
<td>254/2552</td>
<td>-/- fail-3</td>
<td>2713</td>
<td>-/- fail-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-1</td>
<td></td>
<td>247/2992</td>
<td>1.56684</td>
<td>1.34</td>
<td>-/-</td>
<td>fail-3</td>
<td></td>
</tr>
<tr>
<td>10-1</td>
<td></td>
<td>490/6095</td>
<td>5.99059</td>
<td>3.15</td>
<td>7200/20151</td>
<td>6.05138</td>
<td>1800.15</td>
</tr>
<tr>
<td>1-2</td>
<td>LMBM-1P</td>
<td>10389/43599</td>
<td>0.880569</td>
<td>0.227</td>
<td>-/-</td>
<td>fail-3</td>
<td></td>
</tr>
<tr>
<td>2-2</td>
<td></td>
<td>152/841</td>
<td>0.221364</td>
<td>12.66</td>
<td>4295/14597</td>
<td>0.008487</td>
<td>319.57</td>
</tr>
<tr>
<td>3-2</td>
<td></td>
<td>35/306</td>
<td>-/- fail-3</td>
<td>0.12</td>
<td>167/517</td>
<td>-/- fail-3</td>
<td>7.20</td>
</tr>
<tr>
<td>4-2</td>
<td></td>
<td>89/609</td>
<td>2808.45</td>
<td>0.46</td>
<td>462/815</td>
<td>2808.45</td>
<td>30.29</td>
</tr>
<tr>
<td>5-2</td>
<td></td>
<td>26/103</td>
<td>2800.19</td>
<td>0.07</td>
<td>334/700</td>
<td>2796.35</td>
<td>20.29</td>
</tr>
<tr>
<td>6-2</td>
<td></td>
<td>213/1507</td>
<td>2.79690</td>
<td>0.66</td>
<td>591/4036</td>
<td>2.77674</td>
<td>13.76</td>
</tr>
<tr>
<td>7-2</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-2</td>
<td></td>
<td>23/81</td>
<td>4466.99</td>
<td>0.06</td>
<td>82/124</td>
<td>4507.02</td>
<td>2.41</td>
</tr>
<tr>
<td>9-2</td>
<td></td>
<td>450/3062</td>
<td>483.441</td>
<td>1.24</td>
<td>122/480</td>
<td>497.297</td>
<td>3.09</td>
</tr>
<tr>
<td>10-2</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3</td>
<td></td>
<td>-/- fail-2</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td></td>
<td>42/318</td>
<td>0.303804</td>
<td>4.77</td>
<td>91/517</td>
<td>0.007981</td>
<td>8.85</td>
</tr>
<tr>
<td>3-3</td>
<td></td>
<td>85/769</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-3</td>
<td></td>
<td>61/653</td>
<td>2001.63</td>
<td>0.33</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-3</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-3</td>
<td></td>
<td>30/88</td>
<td>0.405473</td>
<td>0.04</td>
<td>26/41</td>
<td>3.04435</td>
<td>0.06</td>
</tr>
<tr>
<td>7-3</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-3</td>
<td></td>
<td>169/1662</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-3</td>
<td></td>
<td>82304/2645587</td>
<td>0.250063</td>
<td>655.14</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-3</td>
<td>LMBM-1P</td>
<td>91/2157</td>
<td>1.85396</td>
<td>0.56</td>
<td>-/-</td>
<td>fail-2</td>
<td></td>
</tr>
<tr>
<td>1-4</td>
<td></td>
<td>9943/164502</td>
<td>0.388891</td>
<td>57.37</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-4</td>
<td></td>
<td>51/963</td>
<td>1.02087</td>
<td>14.48</td>
<td>91/517</td>
<td>0.007981</td>
<td>8.74</td>
</tr>
<tr>
<td>3-4</td>
<td></td>
<td>74/598</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-4</td>
<td></td>
<td>59/689</td>
<td>2001.72</td>
<td>0.41</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-4</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-4</td>
<td></td>
<td>28/63</td>
<td>0.405549</td>
<td>0.04</td>
<td>26/41</td>
<td>3.04435</td>
<td>0.06</td>
</tr>
<tr>
<td>7-4</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-4</td>
<td></td>
<td>194/1569</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-4</td>
<td></td>
<td>203/3168</td>
<td>0.250222</td>
<td>1.12</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-4</td>
<td></td>
<td>208/2713</td>
<td>1.39342</td>
<td>1.28</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-5</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-5</td>
<td></td>
<td>50/599</td>
<td>0.022148</td>
<td>8.97</td>
<td>200/478</td>
<td>0.600211</td>
<td>9.42</td>
</tr>
<tr>
<td>3-5</td>
<td></td>
<td>13/103</td>
<td>-/- fail-3</td>
<td>0.04</td>
<td>172/466</td>
<td>-/- fail-2</td>
<td></td>
</tr>
<tr>
<td>4-5</td>
<td></td>
<td>11/74</td>
<td>4043.82</td>
<td>0.04</td>
<td>352/913</td>
<td>4043.82</td>
<td>30.45</td>
</tr>
<tr>
<td>5-5</td>
<td></td>
<td>11/74</td>
<td>4043.82</td>
<td>0.04</td>
<td>352/947</td>
<td>4043.82</td>
<td>34.22</td>
</tr>
<tr>
<td>6-5</td>
<td></td>
<td>28/302</td>
<td>5.81129</td>
<td>0.17</td>
<td>19/36</td>
<td>5.81129</td>
<td>0.04</td>
</tr>
<tr>
<td>7-5</td>
<td></td>
<td>9/46</td>
<td>589.475</td>
<td>0.09</td>
<td>204/521</td>
<td>589.469</td>
<td>29.60</td>
</tr>
<tr>
<td>8-5</td>
<td></td>
<td>34/471</td>
<td>-/- fail-3</td>
<td>0.17</td>
<td>33/254</td>
<td>-/- fail-3</td>
<td></td>
</tr>
<tr>
<td>9-5</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-5</td>
<td></td>
<td>-/- fail-3</td>
<td>-/- fail-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td>781/7008</td>
<td>4.60</td>
<td>1457/3357</td>
<td>193.74</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
with the new solver. Particularly, it failed to solve problem 2 with the desired accuracy regardless of the constraint used. These kind of difficulties were quite predictable, since problem 2 is reported to be difficult to solve with limited memory bundle method even without constraints [13].

In both bound constrained and inequality constrained problems the numbers of function evaluations used with LMBM–IP were much larger than the numbers of iterations. For instance, in inequality constrained problems LMBM–IP used about nine function evaluations per iteration while with PBNCGC this ratio was only about two (see Table 2). This also means that LMBM–IP usually needed more function evaluations than PBNCGC even if the number of iterations was smaller. The reason for large amount of function evaluations needed with LMBM–IP is that near the solution the feasible search direction $d_k$ supports very short feasible segments and many function evaluations are needed in line search before a feasible step size is obtained (see Algorithm 3.2). The possibilities of avoiding this kind of effect by using the curvilinear search inside the feasible region (see, e.g., [17, 34]) are to be studied.

6 Conclusions

In this paper, we have described a new interior point based limited memory bundle method for inequality constrained nonsmooth optimization. The new solver is suitable for general nonlinearly constrained large-scale problems, where neither the objective function nor the constraints are supposed to be continuously differentiable or convex.

We have studied the convergence properties of the method and given some results from numerical experiments. The preliminary numerical experiments confirm that the limited memory interior point bundle solver is efficient for both convex and nonconvex large-scale nonsmooth inequality constrained optimization problems. With large numbers of variables it used significantly less CPU time than the other solver tested.

Acknowledgements

The authors would like to thank Ph. Lic. Elina Madetoja (University of Kuopio, Finland) for valuable comments and ideas.

Appendix

Limited memory matrices. The limited memory variable metric matrices used in our algorithm are represented in the compact matrix form originally described in [5]. 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

\[
S_k = \begin{bmatrix} s_{k-\hat{m}_k} & \ldots & s_{k-1} \end{bmatrix} \quad \text{and} \quad U_k = \begin{bmatrix} u_{k-\hat{m}_k} & \ldots & u_{k-1} \end{bmatrix},
\]

where the correction pairs \((s_i, u_i), (i < k)\) are obtained in Step 4 of Algorithm 3.1.

The inverse limited memory BFGS update is defined by the formula

\[
D_k = \vartheta_k I + \left[ S_k \vartheta_k U_k \right] \begin{bmatrix} (R_k^{-1})^T (C_k + \vartheta_k U_k^T U_k) R_k^{-1} & -(R_k^{-1})^T \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^T \\ \vartheta_k U_k^T \end{bmatrix},
\]  

(29)

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

\[
(R_k)_{ij} = \begin{cases} (s_{k-\hat{m}_k-1+i})^T (u_{k-\hat{m}_k-1+j}), & \text{if } i \leq j \\ 0, & \text{otherwise,} \end{cases}
\]

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

\[
C_k = \text{diag} [s_{k-\hat{m}_k}^T, u_{k-\hat{m}_k}^T, \ldots, s_{k-1}^T, u_{k-1}^T],
\]

and \( \vartheta_k \) is a positive scaling parameter.

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

\[
D_k = \vartheta_k I - (\vartheta_k U_k - S_k)(\vartheta_k U_k^T U_k - R_k - R_k^T + C_k)^{-1}(\vartheta_k U_k - S_k)^T.
\]  

(30)

The similar representations for the direct limited memory BFGS and SR1 updates can be given (see, e.g., [5]). However, the implementation of our algorithm only needs the inverse update formulae to be used.

**Lemma 6.1.** The condition

\[
-d_j^T u_j - \left( \tilde{\xi}_{f,j} + \sum_{i \in p} \tilde{\mu}_{i,j+1} \tilde{\xi}_{g_{i,j}} \right)^T s_j < 0 \quad \text{for all } j = 1, \ldots, k - 1,
\]  

(31)

where \( \tilde{\mu}_{j+1} = \mu_{j+1} + \rho_j \mu_{j+1}^\beta \), assures the positive definiteness of the matrices obtained by the limited memory SR1 update. Furthermore, it implies \( u_j^T s_j > 0 \) for all \( j = 1, \ldots, k - 1 \) that assures the positive definiteness of the matrices obtained by the limited memory BFGS update.

**Proof.** Using condition (31), the fact that we have \( s_j = t_{R}^j \theta_j d_j, t_{R}^j > 0, \theta_j \in (0, 1], \) and \( \tilde{\xi}_{f,j} + \sum_{i \in p} \tilde{\mu}_{i,j+1} \tilde{\xi}_{g_{i,j}} = -B_j d_j \) for all \( j = 1, \ldots, k - 1, \) we obtain

\[
d_i^T u_j > -t_{R}^j \theta_j d_j^T \left( \tilde{\xi}_{f,j} + \sum_{i \in p} \tilde{\mu}_{i,j+1} \tilde{\xi}_{g_{i,j}} \right) = t_{R}^j \theta_j d_j^T B_j d_j \geq t_{R}^j \theta_j d_j^T B_j' d_j
\]  

(32)

for all \( i = 1, \ldots, k - 1. \) Now, by replacing equation (29) in [14] by (32), the first part of the proof proceeds similar to the proof of Lemma 10 in [14].

The second part of the lemma follows from (32) provided by the positive definiteness of (previous matrices) \( B_j \) and the fact that we have \( s_j = t_{R}^j \theta_j d_j. \)

In our proposal, the individual updates that would violate positive definiteness are skipped (for more details, see [11, 13, 14]).
Constraints. The following constraint functions were used in our inequality constrained experiments:

1. Modification of Broyden tridiagonal constraint

   \[ g_i(x) = (3.0 - 2.0x_{i+1})x_{i+1} - x_i - 2.0x_{i+2} + 1.0, \quad i = 1, \ldots, 5, \]

   for problems 1, 2, 6, 7, 9, and 10 in [13] and

   \[ g_i(x) = (3.0 - 2.0x_{i+1})x_{i+1} - x_i - 2.0x_{i+2} + 2.5, \quad i = 1, \ldots, 5, \]

   for problems 3, 4, 5, and 8 in [13] (for original Broyden tridiagonal constraint, see, e.g., [28]).

2. Modification of Broyden tridiagonal constraint II

   \[ g_1(x) = \sum_{i=1}^{n-2} ((3.0 - 2.0x_{i+1})x_{i+1} - x_i - 2.0x_{i+2} + 1.0), \]

   for problems 1, 2, 6, 7, 9, and 10 in [13] and

   \[ g_1(x) = \sum_{i=1}^{n-2} ((3.0 - 2.0x_{i+1})x_{i+1} - x_i - 2.0x_{i+2} + 2.5), \]

   for problems 3, 4, 5, and 8 in [13].

3. Modification of MAD1

   \[ g_1(x) = \max \left\{ x_1^2 + x_2^2 + x_1x_2 - 1.0, \sin x_1, -\cos x_2 \right\}, \]

   \[ g_2(x) = -x_1 - x_2 + 0.5, \]

   (for original problem, see, e.g., [29]).

4. Modification of MAD1 II

   \[ g_1(x) = x_1^2 + x_2^2 + x_1x_2 - 1.0, \]

   \[ g_2(x) = \sin x_1, \]

   \[ g_3(x) = -\cos x_2, \]

   \[ g_4(x) = -x_1 - x_2 + 0.5. \]

5. Simple modification of MAD1

   \[ g_1(x) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2 + x_ix_{i+1} - 2.0x_i - 2.0x_{i+1} + 1.0) \]
for problems 1, 2, 6, 7, 9, and 10 in [13] and

$$g_1(x) = \sum_{i=1}^{n-1} \left( x_i^2 + x_{i+1}^2 + x_ix_{i+1} - 1.0 \right)$$

for problems 3, 4, 5, and 8 in [13].

In all cases the starting point were chosen to be feasible.

**References**


