Polska
Akademia
Nauk
Instytut
Badań
Systemowych

Methodology and applications of decision support systems

Proceedings of the 3-rd
Polish-Finnish Symposium
Gdańsk-Sobieszewo, September 26-29, 1988

edited by
Roman Kulikowski

Polska Akademia Nauk Instytut Badań Systemowych

Methodology and applications of decision support systems

Proceedings of the 3-rd
Polish-Finnish Symposium
Gdańsk-Sobieszewo, September 26-29, 1988

edited by
Roman Kulikowski

Warszawa 1989

Secretary of the Conference
dr. Andrzej Stachurski

Wykonano z gotowych oryginałów tekstowych
dostarczonych przez autorów

41267

ISBN 83-00-02543-X

Numerical Experiments with Various Nonsmooth Optimization Methods

MARKO M. MÄKELÄ

University of Jyväskylä, Department of Mathematics
Seminaarinkatu 15, SF-40100 Jyväskylä, Finland

Abstract. There exist continuously growing interest in nonsmooth optimization methods in recent years. The aim of this paper is to present the results of numerical experiments with two main classes of nonsmooth programming methods. We have tested Kiwiel's linearization methods and Lemarechal's bundle methods. As test problems we used well-known test problems from litterature: Crescent, Shor's problem, Equil and Ill-contioned LP. In addition we tested two different kind of optimal control problems: Contact and Stefan, which have been studied in University of Jyväskylä. Some corollaries following from the presented results are included. In general we can state, that Kiwiel's methods seem to be more reliable and demand less computer resource than bundle methods.

1. INTRODUCTION

One of the most important stages in decision making process is the mathematical modelling of natural phenomena by using some mathematical tools. The basic idea of mathematical modelling is to simulate some natural process with computer. By simulation it is possible to predict the behavior of the phenomenon or to learn how best to get the desired outcome.

Amazingly often the mathematical model can be formulated as a nonlinear optimization problem

$$(1.1) \qquad \text{minimize} \quad f(x) \quad \text{subject to} \quad F_i(x) \le 0 \quad \text{for} \quad i = 1, \ldots, m,$$

over all $x \in \mathbb{R}^n$. If the problem functions f and F_i are continuously differentiable, then the problem (1.1) is said to be smooth. The most efficient methods and algorithms are generated for the smooth versions of the optimization problem (1.1). These classical algorithms use gradient information of the problem functions and are often based on an iterative process. Given a starting point $x_1 \in \mathbb{R}^n$, an iterative method constructs a sequence (x_i) in $\mathbb{R}^n$ that is intended to converge to the required solution. The basic algorithm for solving the problem (1.1) is as follows.

Algorithm 1.1.

Step 0: *(Initialization).* Set $k = 1$ and find a starting point x_1 in $\mathbb{R}^n$.

Step 1: *(Direction finding).* Find a descent direction d_k for f which is feasible for G at x_k. In other words, find $d_k \in \mathbb{R}^n$ and $\varepsilon > 0$ such that

$$(1.2) \quad f(x_k + td_k) < f(x_k) \quad \text{and} \quad x_k + td_k \in G \quad \text{for all} \quad t \in (0, \varepsilon].$$

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

Step 3: *(Line search).* Find a step length $t_k \in (0, \varepsilon]$ such that

$$(1.3) \quad t_k = \arg\min\{f(x_k + td_k) \mid t \in (0, \varepsilon]\} \quad \text{and} \quad x_k + t_k d_k \in G.$$

Step 4: *(Updating).* Set $x_{k+1} = x_k + t_k d_k$, increase k by 1 and go to Step 1.

For smooth problems Step 1 usually is accomplished with the use of gradient information. However in practice we often get into the situation when the problem (1.1) is nonsmooth, because nature need not behave smoothly. Then we cannot directly use the methods which need gradient information while the usual methods which do not need gradients are often not efficient enough. The easiest way to solve the problem is to generate a smooth approximation problem which can be solved by classical methods. This however causes more difficulties in the form of approximation errors, which can be catastrophic.

Another approach is to use the results of nonsmooth analysis, replacing the gradients by subgradients and subdifferentials. This approach requires, however, that we know the whole subdifferential at every point. In practice this is often too big a requirement and we have to be content with one subgradient at each point.

For these reasons much research has been done during the last two decades to produce new efficient methods for nonsmooth optimization. These methods mainly follows the general ideas of algorithm 1.1 but the nonsmoothness requires additional work at almost every step. The first and hardest problem is in Step 1, to generate the descent direction for the objective function. In smooth analysis the direction opposite of the gradient is locally the steepest descent. For an arbitrary subgradient this is not true anymore and the direction opposite of a subgradient need not even be one of descent. The next difficulty is the line search operation in Step 3. In smooth methods the best way to do this is to use some efficient univariate optimization method, which in general needs derivatives and so is not suitable for nonsmooth optimization. Smooth polynomial interpolation is an efficient method which needs no derivatives, but for nonsmooth functions the approximation may be too rough. Also, the stopping criterion is not clear any more.

2. Derivation of the Methods

At the moment the most promising methods for nonsmooth optimization can be divided into two main classes: linearization methods and bundle methods. The essential difference between these methods is in the way they handle the subgradient information to generate the descent direction in Step 1 of algorithm 1.1. In what follows we give a short derivation of these two method classes and how they generate the descent direction. For simplifying the presentation we restrict our development to the convex unconstrained case. We suppose that we can evaluate one subgradient $\xi \in \partial f(x)$ and the function value $f(x)$ at each point x in $\mathbb{R}^n$. We denote by $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$, respectively , the usual inner product and norm in real Euclidian space $\mathbb{R}^n$. For convex functions the subdifferential at the point y is defined by

$$(2.1)\qquad \partial f(y) = \{\xi \in \mathbb{R}^n \mid f(x) \geq f(y) + \langle \xi, x-y\rangle \quad \text{for all} \quad x \in \mathbb{R}^n\}.$$

The elements $\xi \in \partial f$ are called subgradients.

2.1 Linearization Method.

The main idea in linearization method is to form a piecewise linear approximation to the objective function using the linearizations generated by subgradients at different points. This approximation is naturally easier to handle and the desenct direction for it can be found as a solution of a quadratic optimization problem. After this we can apply the result, which tells that this direction is descent also for the original objective function.

It is known from convex analysis (see [9]) that the convex function f has at each point $x \in \mathbb{R}^n$ a representation

$$(2.2)\qquad f(x) = \max\{f(y) + \langle \xi_y, x-y\rangle \mid \xi \in \partial f(y),\ y \in \mathbb{R}^n\}$$

and by using the linearizations

$$(2.3)\qquad \begin{aligned} f_\xi(x) &= f(y) + \langle \xi, x-y\rangle \quad &&\text{for all} \quad x \in \mathbb{R}^n \quad \text{and} \\ \hat{f}_y(x) &= \max\{f_\xi(x) \mid \xi \in \partial f(y)\} \quad &&\text{for all} \quad x \in \mathbb{R}^n, \end{aligned}$$

we get the representation

$$(2.4)\qquad f(x) = \max\{\hat{f}_y(x) \mid y \in \mathbb{R}^n\} \quad \text{for all} \quad x \in \mathbb{R}^n.$$

It can be proofed that a descent direction for linearization $\hat{f}_y$ is a descent direction also for f. Notice however, that for the representation (2.4) we need the whole subdifferential $\partial f(y)$. For this reason we can't use linearizations (2.3) as such, but we will construct approximate versions of them as follows. We suppose that also besides in addition the current iteration point x_k we have some auxiliary points y_j in $\mathbb{R}^n$ and subgradients ξ_j in $\partial f(y_j)$ for $j \in J_k$, where the index set J_k is a nonempty subset of the set $\{1, \ldots, k\}$. Then the approximate versions of (2.3) can be defined by

$$\text{(2.5)} \qquad \begin{aligned} & f_j(x) = f_{\xi_j}(x) = f(y_j) + \langle \xi_j, x - y_j \rangle \qquad \text{for all} \quad j \in J_k \quad \text{and} \\ & \hat{f}^k(x) = \max\{f_j(x) \mid j \in J_k\}. \end{aligned}$$

This approximation function $\hat{f}^k$ is now a suitable piecewise linear approximation to the objective function f, which can be used in practice. Our aim now is to find a descent direction for $\hat{f}^k$. To quarantee the uniqness of the solution we use. a regularizing penalty term $\frac{1}{2}\|d\|^2$ and so we get the problem

$$\text{(2.6)} \qquad \text{minimize} \quad \hat{f}^k(x_k + d) + \tfrac{1}{2}\|d\|^2 \quad \text{over all} \quad d \in \mathbb{R}^n,$$

which is equivalent to find the multipliers λ_j for $j \in J_k$ such that they solve the quadratic dual problem

$$\text{(2.7)} \qquad \begin{aligned} & \text{minimize} \quad \tfrac{1}{2}\Big\|\sum_{j \in J_k} \lambda_j \xi_j\Big\|^2 - \sum_{j \in J_k} \lambda_j f_j^k \\ & \text{subject to} \quad \sum_{j \in J_k} \lambda_j = 1 \quad \text{and} \quad \lambda_j \geq 0 \quad \text{for all} \quad j \in J_k, \end{aligned}$$

where $f_j^k = f_j(x_k)$ for $j = 1, \ldots, k$ and then the direction $d_k = -\sum_{j \in J_k} \lambda_j^k \xi_j$..

2.2 Bundle Method.

The guiding principle behind bundle methods is to use the theory of ε-subdifferentials, which are enlargements of the conventional subdifferentials. The same subgradients at different points as before are now collected at one point and the convex hull of this subgradient bundle is taken as an approximating ε-subdifferential at this point. Then we can make use of the result that the best subgradient of the subdifferential minimizes the norm. This subgradient can also be found as a solution of a certain quadratic optimization problem which resembles considerably the one in the linearization method.

It is known from convex analysis (see [9]) that the convex function f has the directional derivate $f'(x; d)$ at $x \in \mathbb{R}^n$ in each direction $d \in \mathbb{R}^n$, where

$$\text{(2.8)} \qquad f'(x; d) = \lim_{t \downarrow 0} \frac{f(x + td) - f(x)}{t}.$$

This directional derivative is a measure for the descent we can expect along $x + td$. This leads to the descent direction finding problem

$$\text{(2.9)} \qquad \begin{array}{ll} \text{minimize} & f'(x;d) \\ \text{subject to} & \|d\| \le 1. \end{array}$$

The additional constraint $\|d\| \le 1$ in (2.9) becomes necessary, since $f'(x;\cdot)$ is positively homogenous. By using the identity

$$\text{(2.10)} \qquad f'(x;d) = \max\{\langle \xi, d\rangle \mid \xi \in \partial f(x)\}$$

we rewrite (2.9) in the form

$$\text{(2.11)} \qquad \min_{\|d\|\le 1} \max_{\xi\in\partial f(x)} \langle \xi, d\rangle.$$

Hence, by a well-known Minimax Theorem, this is equivalent to

$$\text{(2.12)} \qquad \max_{\xi\in\partial f(x)} \min_{\|d\|\le 1} \langle \xi, d\rangle.$$

For given $\xi \in \partial f(x)$ the solution of the latter minimizing problem is $d = -\xi/\|\xi\|$, so we have

$$\text{(2.13)} \qquad \max_{\xi\in\partial f(x)} \min_{\|d\|\le 1} \langle \xi, d\rangle = \max_{\xi\in\partial f(x)} \langle \xi, -\xi/\|\xi\|\rangle = -\max_{\xi\in\partial f(x)} \|\xi\|.$$

Hence, for solving (2.9) we have to study the minimum-norm problem (which is uniquely solvable since $\partial f(x)$ is a nonempty closed convex set)

$$\text{(2.14)} \qquad \min_{\xi\in\partial f(x)} \|\xi\|.$$

Notice, that for the representation (2.14) we need again the whole subdifferential $\partial f(x)$ and for this reason we have to approximate it somehow. The bundle methods are based on the theory of ε-subdifferentials, which in convex case is defined for $\varepsilon > 0$ by

$$\text{(2.15)} \qquad \partial_\varepsilon f(y) = \{\xi \in \mathbb{R}^n \mid f(x) \ge f(y) + \langle \xi, x - y\rangle - \varepsilon \quad \text{for all} \quad x \in \mathbb{R}^n\}.$$

Notice, that for $\varepsilon = 0$ this coincides with the ordinary subdifferential. By using the same linearizations f_j as before we define the bundle $G_k(\varepsilon)$ at the iteration point x_k as follows

$$\text{(2.16)} \quad G_k(\varepsilon) = \{\xi \in \mathbb{R}^n \mid \xi = \sum_{j\in J_k} \lambda_j \xi_j,\ \sum_{j\in J_k} \lambda_j \alpha_j^k \le \varepsilon,\ \lambda_j \ge 0,\ \sum_{j\in J_k} \lambda_j = 1\},$$

where $\alpha_j^k = f(x_k) - f_j^k$ is the linearization error. It is easy to prove that $G_k(\varepsilon)$ is a convex compact set and $G_k(\varepsilon) \subset \partial_\varepsilon f(x_k)$. Now we replace $\partial f(x_k)$ by the bundle $G_k(\varepsilon)$ in (2.14) and we get the problem

$$\min_{\xi \in G_k(\varepsilon)} \|\xi\|, \tag{2.17}$$

which can be modified in the form

$$\text{minimize} \quad \tfrac{1}{2}\|\xi\|^2 \quad \text{over all} \quad \xi \in G_k(\varepsilon). \tag{2.18}$$

This is equivalent to find the multipliers λ_j for $j \in J_k$ such that they solve the quadratic dual problem

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\Big\| \sum_{j\in J_k} \lambda_j \xi_j \Big\|^2 \\ &\text{subject to} && \sum_{j \in J_k} \lambda_j = 1, \\ & && \sum_{j\in J_k} \lambda_j \alpha_j^k \le \varepsilon_k \\ &\text{and} && \lambda_j \ge 0 \quad \text{for all} \quad j \in J_k \end{aligned} \tag{2.19}$$

and then the direction $d_k = -\sum_{j\in J_k} \lambda_j^k \xi_j$.

3. Numerical Examples from Litterature

This chapter is devoted for testing the methods presented in previous chapters in practice. The main idea is to compare the linearization methods with the bundle methods. For each test problem we choose a suitable algorithm of both method classes. All the algorithms were implemented in FORTRAN on a VAX 8600 computer with the relative accuracy of 10^{-16} in double precision (sixteen-digits precision). To solve the quadratic linearly constrained smooth optimization problem in direction finding of all the algorithms we used the algorithm E04NAF from NAG-subroutine library.

The following algorithms were used:

Linearization Methods.

SUCOUC -	the SUbgradient method for COnvex UnConstrained optimization
SSCOLC -	the Special Subgradient method for COnvex Linearly Constrained optimization
SLNCUC -	the method with Subgradient Locality measure for NonConvex UnConstrained optimization
SSNCLC -	the Special method with Subgradient locality measure for NonConvex Linearly Constrained optimization.

Bundle Methods.

BUCOUC* - the BUndle method for COnvex UnConstrained optimization

BLNCUC - the Bundle method with subgradient Locality measure for Non-Convex UnConstrained optimization

M2FC1 - the bundle algorithm due to Lemarechal.

Other Methods.

E04VC - the sequential quadratic programming algorithm from NAG-subroutine library (for smooth optimization).

The following abbreviations will be used:

	it	—	the number of iterations
	nf	—	the number of function and subgradient calls
	CPU	—	the CPU time in seconds
(3.1)	stop	—	the iteration termination reason
	x	—	the final solution
	$f(x)$	—	the final value of the objective function
	$\|\|x - x^*\|\|$	—	the norm of error.

3.1 Crescent.

The first test problem is to

$$\text{(3.2)} \quad \text{minimize} \quad f(x) = \max\{g_1(x),\ g_2(x)\} \quad \text{over all} \quad x = (x_1, x_2) \in \mathbb{R}^2,$$

where

$$\text{(3.3)} \quad \begin{cases} g_1(x_1,x_2) = x_1^2 + (x_2 - 1)^2 + x_2 - 1 \\ g_2(x_1,x_2) = -x_1^2 - (x_2 - 1)^2 + x_2 + 1. \end{cases}$$

This is the test problem 2.4 of Kiwiel [2] and the second problem of Kiwiel [3]. The objective function is highly nonconvex and has narrow crescent-shaped level sets which force algorithms to make very short steps. It is easy to see that the optimum point is $x^* = (0,0)$, where $f(x^*) = 0$. The subgradient can be calculated by

$$\text{(3.4)} \quad \partial f(x) = \begin{cases} \partial g_1(x) & \text{when} \quad g_1(x) > g_2(x) \\ \text{conv}\{\partial g_1(x) \cup \partial g_2(x)\} & \text{when} \quad g_1(x) = g_2(x) \\ \partial g_2(x) & \text{when} \quad g_1(x) < g_2(x). \end{cases}$$

Our starting point $x^1 = (-1.5, 2)$ has $f(x^1) = 4.25$. In spite of the nonconvex objective function, we tested also the algorithms SUCOUC and BUCOUC, which seem to work well in this case. The results for various algorithms are given in Table 3.1.

Algorithm	it	nf	CPU	x_1	x_2	$f(x)$
SUCOUC	4	9	0.10	0.0	0.0	0.0
SLNCUC	7	23	0.10	0.0	$-3.112 \cdot 10^{-6}$	$3.112 \cdot 10^{-6}$
SLNCUC	10	29	0.12	0.0	$-2.469 \cdot 10^{-12}$	$2.469 \cdot 10^{-12}$
BUCOUC	6	19	0.10	0.0	$-7.894 \cdot 10^{-4}$	$7.900 \cdot 10^{-4}$
BLNCUC	11	49	0.13	0.0	$-5.315 \cdot 10^{-8}$	$5.315 \cdot 10^{-8}$
BLNCUC	11	49	0.13	0.0	$-5.315 \cdot 10^{-8}$	$5.315 \cdot 10^{-8}$

Table 3.1

In Figure 3.2 we see the generation of the solution path of algorithm SUCOUC. The paths of the other algorithms were nearly same; there occurred only a bit differences nearby the solution point.

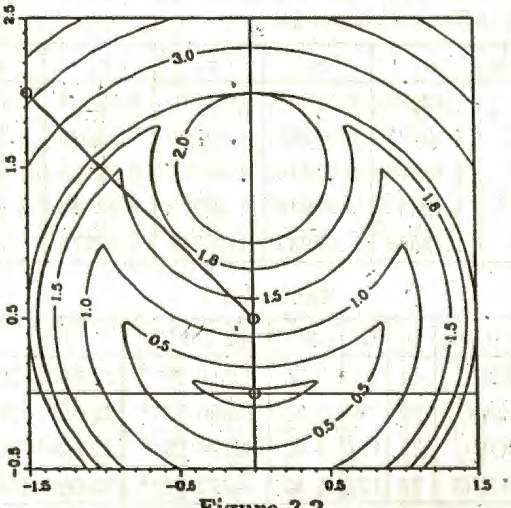

Figure 3.2

3.2 Shor.

The Shor's test problem (test problem 2.1 of Kiwiel [2]) is to

(3.5) minimize $f(x) = \max\{g_i(x) \mid i = 1, \dots, 10\}$ over all $x \in \mathbb{R}^5$, where

$$g_i(x) = b_i \sum_{j=1}^{5} (x_j - A_{ij})^2, \qquad \text{for} \quad i = 1, \ldots, 10. \tag{3.6}$$

The matrix A and the vector b are defined by

$$A^T = \begin{pmatrix} 0 & 2 & 1 & 1 & 3 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 2 & 4 & 2 & 2 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 2 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 3 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{3.7}$$

$$b = (1.0, 5.0, 10.0, 2.0, 4.0, 3.0, 1.7, 2.5, 6.0, 3.5).$$

The problem (3.5) is convex and the optimal solution is

$$x^* = (1.12434,\ 0.97945,\ 1.47770,\ 0.92023,\ 1.12429), \tag{3.8}$$

where $f(x^*) = 22.60016$. We used the starting point $x^1 = (0, 0, 0, 0, 1)$, where $f(x^1) = 80$. The results for various algorithms for the accuracy parameter $\varepsilon = 10^{-6}$ are given in Tables 3.3 and 3.4

Algorithm	x_1	x_2	x_3	x_4	x_5
x^*	1.12434	0.97945	1.47770	0.92023	1.12429
SUCOUC	1.12433	0.97945	1.47760	0.92023	1.12428
SLNCUC	1.12436	0.97946	1.47770	0.92025	1.12429
BUCOUC	1.12437	0.97946	1.47773	0.92025	1.12429
BLNCUC	1.12664	0.97932	1.48158	0.92275	1.12410

Table 3.3

Algorithm	it	nf	CPU	$\|x - x^*\|$	$f(x)$
SUCOUC	44	83	2.70	$1.010 \cdot 10^{-4}$	22.60016255
SLNCUC	29	76	1.30	$3.000 \cdot 10^{-5}$	22.60016220
BUCOUC	35	177	3.90	$4.796 \cdot 10^{-5}$	22.60016259
BLNCUC	27	127	3.40	$5.172 \cdot 10^{-3}$	22.60028332

Table 3.4

3.3 Equil.

The next test problem is the problem number 3 of Lemarechal and Mifflin [6]. This problem is a nonconvex min-max formulation of an economic equilibrium problem and it is to

$$\text{(3.9)}\quad \begin{aligned} &\text{minimize} \quad f(x) = \max\{f_i(x) \mid i = 1,\ldots,8\} \\ &\text{subject to} \quad \sum_{j=1}^{8} x_j = 1 \quad \text{and} \quad x_j \geq 0 \quad \text{for} \quad j = 1,\ldots,8, \end{aligned}$$

where

$$\text{(3.10)}\quad \begin{aligned} f_i(x) &= \sum_{k=1}^{5} f_i^k \qquad \text{for} \quad i = 1,\ldots,8, \\ f_i^k(x) &= \frac{\left(A_{ki}\sum_{j=1}^{8} W_{kj}x_j\right)}{\left(x_i^{d_k}\sum_{j=1}^{8} A_{kj}x_j^{1-d_k}\right)} - W_{ki} \quad \text{for} \quad k = 1,\ldots,5. \end{aligned}$$

The data is given by

$$\text{(3.11)}\quad \begin{aligned} W &= \begin{pmatrix} 3.0 & 1.0 & 0.1 & 0.1 & 5.0 & 0.1 & 0.1 & 6.0 \\ 0.1 & 10.0 & 0.1 & 0.1 & 5.0 & 0.1 & 0.1 & 0.1 \\ 0.1 & 9.0 & 10.0 & 0.1 & 4.0 & 0.1 & 7.0 & 0.1 \\ 0.1 & 0.1 & 0.1 & 10.0 & 0.1 & 3.0 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 11.0 \end{pmatrix} \\ A &= \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 2.0 & 0.8 & 1.0 & 0.5 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.2 & 0.8 & 1.2 & 1.6 & 2.0 & 0.6 & 0.1 \\ 2.0 & 0.1 & 0.6 & 2.0 & 1.0 & 1.0 & 1.0 & 2.0 \\ 1.2 & 1.2 & 0.8 & 1.0 & 1.2 & 0.1 & 3.0 & 4.0 \end{pmatrix} \\ d &= (0.5,\ 1.2,\ 0.8,\ 2.0,\ 1.5). \end{aligned}$$

If we write the linear constraints in form $Cx \leq b$ we get

$$\text{(3.12)}\quad \begin{aligned} C &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \\ b &= (1,\ -1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0). \end{aligned}$$

We used a starting point $x_j = 0.125$ for all $j = 1, \dots, 8$. By Lemarechal and Mifflin [6] the optimal solution x^* is a point near

(3.13) $$x = (0.27,\ 0.03,\ 0.06,\ 0.09,\ 0.07,\ 0.31,\ 0.10,\ 0.07)$$

and we have $f(x^*) = 0$. The main difficulty in this problem is the fact that it doesn't satisfy the Slater constraint qualification (see [2]).

One way to solve this difficulty is to transform this linearly constrained problem to unconstrained problem by using a projected subgradient (see [6] p. 160). To solve this new unconstrained problem we can use the algorithms for unconstrained optimization. The results for accuracy parameter $\varepsilon = 10^{-4}$ are given in Tables 3.5 and 3.6.

	SSCOLC	SSNCLC	SUCOUC	SLNCUC	BUCOUC	BLNCUC
x_1	0.27123571	0.27123609	0.27123531	0.28098236	0.27318287	0.27122362
x_2	0.02956531	0.02956520	0.02956531	0.03062800	0.02977756	0.02956627
x_3	0.06293780	0.06293779	0.06293780	0.06520004	0.06338961	0.06293961
x_4	0.09309016	0.09309012	0.09309019	0.09643621	0.09375841	0.09310999
x_5	0.06723419	0.06723416	0.06723420	0.06965085	0.06771687	0.06724069
x_6	0.30590078	0.30590061	0.30590110	0.31689668	0.30809711	0.30587857
x_7	0.10436453	0.10436454	0.10436454	0.10811583	0.10511379	0.10436640
x_8	0.06567152	0.06567148	0.06567154	0.06803202	0.06614302	0.06567485

Table 3.5

Algorithm	it	nf	CPU	$f(x)$
SSCOLC	72	823	14.80	$3.612 \cdot 10^{-7}$
SSNCLC	101	323	22.50	$7.110 \cdot 10^{-5}$
SUCOUC	47	333	4.00	$4.821 \cdot 10^{-6}$
SLNCUC	39	119	3.40	$2.619 \cdot 10^{-5}$
BUCOUC	61	523	7.70	$7.854 \cdot 10^{-6}$
BLNCUC	36	353	5.70	$2.938 \cdot 10^{-4}$

Table 3.6

3.4 Ill-conditioned LP.

The next problem is a test problem 2.3 of Kiwiel [2]. It is linear programming problem

(3.14) $$\begin{aligned} &\text{minimize} \quad f(x) = \langle c, x \rangle \\ &\text{subject to} \quad Ax \le b \quad \text{and} \quad x_j \ge 0 \quad \text{for} \quad j = 1, \dots, 15, \end{aligned}$$

where

$$(3.15)\qquad \begin{aligned} A_{ij} &= \frac{1}{i+j} && \text{for} \quad i,j = 1,\ldots,15,\\ b_i &= \sum_{j=1}^{15} A_{ij} && \text{for} \quad i = 1,\ldots,15,\\ c_i &= \frac{-1}{1+i} - b_i && \text{for} \quad i = 1,\ldots,15. \end{aligned}$$

This problem is ill-conditioned, since the matrix A is essentially a section of the Hilbert matrix. We used a starting point $x_j = 0$ for all $j = 1,\ldots,15$ and the optimal solution $x_j^* = 1$ for all $j = 1,\ldots,15$, where $f(x^*) = -20.0420$. The main difficulty in this problem is the great number of constraints. For this reason it proved to be better to solve this linearly constrained problem by minimizing the exact penalty function

$$(3.16)\qquad \tilde{f}(x) = \langle c, x\rangle + \rho F(x)_+$$

over all $x \in \mathbf{R}^{15}$, where $\rho = 2n = 30$ is the penalty coefficient. The results for accuracy parameter $\varepsilon = 10^{-6}$ are given in Tables 3.7 and 3.8.

Algorithm	it	nf	CPU	$\|\|x - x^*\|\|$	$f(x)$
SUCOUC	34	41	3.20	$4.099\cdot10^{-2}$	-20.04197718
SLNCUC	33	65	4.30	$4.080\cdot10^{-2}$	-20.04197729
BUCOUC	66	252	4.40	$4.259\cdot10^{-2}$	-20.04197569
BLNCUC	74	295	9.50	$1.288\cdot10^{-1}$	-20.04081915

Table 3.7

	SUCOUC	SLNCUC	BUCOUC	BLNCUC
x_1	0.99658489	0.99661376	0.99655763	1.03073079
x_2	1.01621615	1.01608719	1.01664715	0.94494244
x_3	0.99300648	0.99306986	0.99242187	0.96362071
x_4	0.98595504	0.98604655	0.98540523	0.99431275
x_5	0.98971496	0.98976632	0.98957233	1.01816946
x_6	0.99723803	0.99724183	0.99748536	1.03236845
x_7	1.00453217	1.00450193	1.00502555	1.03795150
x_8	1.00976917	1.00972174	1.01035199	1.03669554
x_9	1.01230774	1.01225740	1.01284939	1.03026966
x_{10}	1.01209942	1.01205663	1.01250321	1.02004348
x_{11}	1.00936681	1.00933851	1.00956676	1.00708600
x_{12}	1.00443696	1.00442733	1.00439174	0.99221369
x_{13}	0.99765832	0.99766952	0.99734514	0.97604379
x_{14}	0.98936154	0.98939436	0.98877106	0.95904057
x_{15}	0.97984316	0.97989743	0.97897546	0.94155207

Table 3.8

4. Numerical Examples from Optimal Control

In addition we tested two different kind of optimal control problems: Contact and Stefan, which have been studied in University of Jyväskylä.

4.1 Contact.

In this example we are going to test three computer codes in solving a test problem. The codes are: the sequential quadratic programming algorithm E04VCE for smooth nonlinear optimization from NAG-subroutine library due to Gill et al., the bundle algorithm M2FC1 due to Lemarechal and our own algorithm SUCOUC.

We consider the following finite dimensional optimal control problem

$$\text{(4.1)} \qquad \underset{u \in \mathbb{R}^m}{\text{minimize}} \, \{E(u, x(u)) = E_1(x(u)) + E_2(u)\},$$

where $x(u)$ is given as the solution of the variational inequality

$$\text{(4.2)} \quad x(u) \in K : \quad \langle Ax(u), y - x(u) \rangle \geq \langle f(u), y - x(u) \rangle \qquad \text{for all} \quad y \in K.$$

Here A is a symmetric, positive definite, $n \times n$ matrix, $f : \mathbb{R}^m \to \mathbb{R}^n$ and $K = \{x \in \mathbb{R}^n \mid x_i \geq c_i\}$ nonempty closed convex set.

The problem (4.1) can be viewed as the discretization of for example the following infinite dimensional optimal control problem

$$\text{(4.3)} \qquad \underset{u \in U}{\text{minimize}} \; \{E(u) = \int_\Omega (y(u) - \varphi)\, dx + \frac{1}{2} \int_\Omega u^2\, dx\},$$

where $y(u)$ is given by

$$\text{(4.4)} \quad y \in \hat{K} : \quad \int_\Omega y(u)'(v' - y(u)')\, dx \geq \int_\Omega f(u)(v - y(u))\, dx \quad \text{for all} \quad v \in \hat{K}.$$

Here $\hat{K} = \{v \in H_0^1(\Omega) \mid v \geq \varphi$ almost everywhere in $\Omega\}$, U is a Banach space and $f : U \to L^2(\Omega)$.

This is the model of an elastic string which is deflected by the force u. The string cannot overpass the obstacle φ. The aim is to maximize the contact between the string and an obstacle with minimum total force.

The problem (4.1) is neither smooth nor convex. In general case, where the cost function is defined implicitly by the state inequality, the computation of subgradient is almost impossible. However in our special case this can be done by using a special algorithm (see [8]).

We have chosen $n = m = 10$, $f(u) = u$, $c = (-3, -3, \ldots, -3)$ and

$$\text{(4.5)} \qquad A_{i,j} = \begin{cases} -1, & |i - j| = 1 \\ 2, & i = j \\ 0, & \text{otherwise.} \end{cases}$$

The problem can be thought as the discretization of the problem (4.3) with FEM (Finite Element Method) or FD (Finite Differences) methods with $\Omega = (0, 12)$ and $h = 1$. The state and adjoint inequalities are solved using the successive overrelaxation method with projection. The iteration is stopped when

$$\text{(4.6)} \qquad \sum_i |x_i^{n+1} - x_i^n| \leq \delta\Big(\sum_i |x_i^{n+1}|\Big),$$

where $\delta > 0$ is a tolerance parameter given by the user. The set of active indices $\tilde{I} = I_a \cup I_s$ is determined by

$$\text{(4.7)} \qquad i \in \tilde{I} \quad \text{iff} \quad |x_i - c_i| \leq \delta |x_i|.$$

Note, that we don't know the exact solution. In our first example the tolerance parameter values were $\varepsilon = 10^{-5}$ and $\delta = 10^{-6}$. The starting point was chosen to be $u^1 = 0$. The results for various algorithms are given in Tables 4.1 and 4.2.

Algorithm	u_1	u_2	u_3	u_4	u_5
E04VCE	-1.0067	-0.9966	-0.0125	-0.0082	-0.0239
M2FC1	-0.9998	-1.0000	-0.0003	-0.0000	-0.0002
SUCOUC	-0.9987	-0.9991	-0.0013	-0.0015	-0.0016
Algorithm	u_6	u_7	u_8	u_9	u_{10}
E04VCE	-0.0239	-0.0082	-0.0125	-0.9966	-1.0067
M2FC1	-0.0002	-0.0000	-0.0002	-1.0000	-0.9998
SUCOUC	-0.0016	-0.0015	-0.0013	-0.9991	-0.9987

Table 4.1

Algorithm	it	nf	stop	$E(u)$
E04VCE	17	70	*fail*	4.000844
M2FC1	23	53	*o.k.*	3.999997
SUCOUC	9	18	*o.k.*	4.000007

Table 4.2

In the second example we take $\varepsilon = \delta = 10^{-6}$ and the starting point

$$u^1 = (-3, 0, 0, 0, 0, 0, 0, 0, 0, -3).$$

At this point $I = I_s$. It is easy to verify that $x_j'(u^1; e^i) \neq x_j'(u^1; -e^i)$ for all i and j, where e^i denotes the i:th coordinate vector. Therefore the cost functional is also nondifferentiable at u^1. The results for various algorithms are given in Tables 4.3 and 4.4.

Algorithm	u_1	u_2	u_3	u_4	u_5
E04VCE	-0.9934	-1.0033	-0.0317	-0.0368	-0.0214
M2FC1	-0.9999	-1.0000	-0.0002	-0.0002	-0.0002
SUCOUC	-1.0013	-0.9990	-0.0007	-0.0001	-0.0002
Algorithm	u_6	u_7	u_8	u_9	u_{10}
E04VCE	-0.0007	-0.0227	-0.0389	-1.0039	-0.9913
M2FC1	-0.0002	-0.0002	-0.0002	-1.0000	-0.9999
SUCOUC	-0.0002	-0.0001	-0.0007	-0.9990	-1.0013

Table 4.3

Algorithm	it	nf	stop	$E(u)$
E04VCE	24	109	*fail*	4.002490
M2FC1	12	34	*o.k.*	3.999999
SUCOUC	15	36	*o.k.*	4.000003

Table 4.4

In the last example the state problem is solved inaccurately i.e. we put $\delta = \varepsilon = 10^{-4}$ and the starting point is $u^1 = 0$. The results for this example are given in Tables 4.5 and 4.6.

Algorithm	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
E04VCE	-0.9554	-0.9280	-0.0749	-0.0026	-0.0028	-0.0028	-0.0026	-0.0749	-0.9280	-0.9554
M2FC1	-0.9986	-1.0006	-0.0021	-0.0127	-0.0136	-0.0136	-0.0127	-0.0021	-1.0006	-0.9985
SUCOUC	-0.9975	-0.9950	-0.0077	-0.0002	-0.0002	-0.0002	-0.0002	-0.0077	-0.9950	-0.9974

Table 4.5

Algorithm	it	nf	stop	$E(u)$
E04VCE	9	30	*fail*	4.012631
M2FC1	9	67	*fail*	4.001035
SUCOUC	6	17	*o.k.*	4.000059

Table 4.6

As we can see the special algorithms for nonsmooth problems are well suited for numerical solution of control problems. However standard software for smooth problems can also be used with caution if one has not nonsmooth software available or one is satisfied with modest accuracy. Due to the quadratic part in the cost functional one cannot expect in the control variables more than half of the correct digits in the value of the cost functional.

4.2 Stefan.

The second optimal control problem is so called Stefan two-phase problem, which is the following boundary control problem

$$(4.8)\quad \underset{u\in U_{ad}}{\text{minimize}}\ F(u) = \int_0^T \left\{ \frac{1}{2}\|y(x,t) - y_d(x,t)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(x,t)\|^2_{L^2(\Gamma)} \right\} dt$$

subject to

$$(4.9)\quad \begin{aligned} &\frac{\partial}{\partial t}v(x,t) - \Delta y(x,t) = f(x,t) && \text{in } Q \\ &v(x,t) \in \beta(y(x,t)) && \text{in } Q \\ &\frac{\partial}{\partial n}y(x,t) = u(x,t) && \text{in } \Sigma \\ &v(x,0) = v^0(x) && \text{in } \Omega, \end{aligned}$$

where $T = 1$, $\Omega = (0,1)\times(0,1)$, $\Gamma = \partial\Omega$, $Q = \Gamma \times \{0,T\}$ is a cylinder with lateral face Σ, $\lambda = \frac{1}{20}$, $U_{ad} = \mathbb{R}$, $y_d \equiv 0$ and the functions are defined by

$$\beta(y) = \begin{cases} y & \text{when } y < 0 \\ [0,2] & \text{when } y = 0 \\ 4y+2 & \text{when } y > 0 \end{cases}$$

$$(4.10)\quad f(x,t) = \begin{cases} 8(2e^{-2t} - 1) & \text{when } x_1^2 + x_2^2 \geq e^{-2t} \\ 2(2e^{-2t} - 2) & \text{when } x_1^2 + x_2^2 \leq e^{-2t} \end{cases}$$

$$v^0 = \beta(y^0)$$

$$y^0(x) = \begin{cases} x_1^2 + x_2^2 - 1 & \text{when } x_1^2 + x_2^2 < 1 \\ 2(x_1^2 + x_2^2 - 1) & \text{when } x_1^2 + x_2^2 > 1 \end{cases}$$

and the starting point is given by

$$(4.11)\quad u^1 = u(x,0) = \begin{cases} 0 & \text{on the axes,} \\ 4 & \text{on the parallels to the axes.} \end{cases}$$

After discretization by FEM we get the nonsmooth unconstrained convex optimization problem, which dimension is $n = 308$ (see [5]). We don't know the exact solution but note, that the optimal value of the cost function F is a strict positive value nearby 0. This problem were computed by three algorithm: M2FC1 (Lemarechal), SUCOUC and BUCOUC. The numerical results for accuracy parameter $\varepsilon = 10^{-3}$ are given in Table 4.7.

Algorithm	it	nf	CPU	$F(u)$
M2FC1	7	21	154.50	0.2320173
SUCOUC	9	9	35.30	0.3268202
BUCOUC	6	11	47.50	0.2646014

Table 4.7

5. Conclusions

Due to these test problems we can do some conclusions. If we compare these two method classes we can say, that the linearization methods were clearly better: they worked more reliable and demanded in general less computer resource. However, in some problems bundle merthods needed less iterations, but they seemed always need more function calls and CPU time.

If we compare single algorithms we can recommend the algorithm SLNCUC. In the optimal control problems our methods needed less resources than the bundle algorithm M2FC1 due to Lemarechal. As a summary we can state that the special algorithms for nonsmooth optimization worked better than the NAG-routine E04VC for smooth optimization.

References

1. Clarke F. H., "Optimization and Nonsmooth Analysis," Wiley-Interscience, New York, 1983.
2. Kiwiel K. C., "Methods of Descent for Nondifferentiable Optimization," Lecture Notes in Mathematics 1133, Springer-Verlag, Berlin, 1985.
3. Kiwiel K. C., *An aggregate subgradient method for nonsmooth and nonconvex minimization*, Journal of Computational and Applied Mathematics 14 (1986), 391–400.
4. Kiwiel K. C., *A method of linearizations for linearly constrained nonconvex nonsmooth minimization*, Mathematical Programming 34 (1986), 175–187.
5. Laitinen E., "Numerical Approximation of Heat Transfer Problems," Licentiate Thesis, University of Jyväskylä, Department of Mathematics, 1986.
6. Lemarechal C. and Mifflin R. (Eds.), "Nonsmooth Optimization," Proceedings of a IIASA Workshop, Pergamon Press, Oxford, 1977.
7. Mäkelä M., "Nonsmooth Optimization: Theory and Numerical Applications," Licentiate Thesis, University of Jyväskylä, Department of Mathematics, 1988.
8. Mäkinen R., "State Constrained Control Problems: Theory and Applications," Licentiate Thesis, University of Jyväskylä, Department of Mathematics, 1987.
9. Rockafellar R. T., "Convex Analysis," Princeton University Press, Princeton, New Yersey, 1970.
10. Rockafellar R. T., "The Theory of Subgradients and its Applications to Problems of Optimization. Convex and Nonconvex Functions," Heldermann Verlag, Berlin, 1981.
11. Shor N. Z., "Minimization Methods for Non-differentiable Functions," Springer-Verlag, Berlin, 1985.
12. Zove J., *Nondifferentiable optimization*, Computational Mathematical Programming (1985), 323–356.