SOLUTION OF FEASIBILITY PROBLEMS VIA NON-SMOOTH OPTIMIZATION

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING AND THE INSTITUTE OF ENGINEERING AND SCIENCES OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Iraci Ouveysi
December, 1990

SOLUTION OF FEASIBILITY PROBLEMS VIA NON-SMOOTH OPTIMIZATION

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

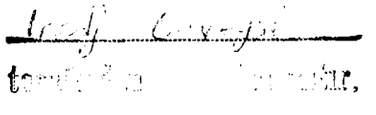
FOR THE DEGREE OF

MASTER OF SCIENCE

By

Iradj Ouveysi

December, 1990



I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Associate Prof. Dr. Osman Oğuz (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Associate Prof. Dr. Mustafa Akgül

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Halim Doğrusöz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Akif Eyler

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Associate Prof. Dr. Péter Kas

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

SOLUTION OF FEASIBILITY PROBLEMS VIA NON-SMOOTH OPTIMIZATION

Iradj Ouveysi
M.S. in Industrial Engineering
Supervisor: Associate Prof. Dr. Osman Oğuz
December, 1990

In this study we present a penalty function approach for linear feasibility problems. Our attempt is to find an effective algorithm based on an exterior method. Any given feasibility (for a set of linear inequalities) problem, is first transformed into an unconstrained minimization of a penalty function, and then the problem is reduced to minimizing a convex, non-smooth, quadratic function. Due to non-differentiability of the penalty function, the gradient type methods can not be applied directly, so a modified nonlinear programming technique will be used in order to overcome the difficulties of the break points. In this research we present a new algorithm for minimizing this non-smooth penalty function.

By dropping the nonnegativity constraints and using conjugate gradient method we compute a maximum set of conjugate directions and then we perform line searches on these directions in order to minimize our penalty function. Whenever the optimality criteria is not satisfied and the improvements in all directions are not enough, we calculate the new set of conjugate directions by conjugate Gram Schmit process, but one of the directions is the element of subdifferential at the present point.

Keywords: Non-Smooth Optimization, Nonlinear Programming, Linear Programming, Feasibility Problem, Penalty Function.

ÖZET

"FEASIBILITY" PROBLEMLERİNİN ÇÖZÜMÜNDE YENİ BİR CEZA FONKSİYONU METODU

Iradj Ouveysi
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Osman Oğuz
Aralık, 1990

Bu çalışmada, doğrusal uyumluluk (feasibility) problemleri için, ceza fonksiyonu yöntemine dayalı yeni bir yaklaşım sunuyoruz. Amacımız, dışsal metod ilkesiyle çalışan etkili bir algoritma bulmaktır. Geliştirilen bu yaklaşımda, ilk olarak, herhangi bir doğrusal uyumluluk problemi (bir lineer eşitsizlikler kümesi için) , sınırlandırılmamış bir ceza fonksiyonunun minimizasyonu problemine dönüştürülür. Bunun sonunda ; dışbükey, kırıklı (non-smooth) ve ikinci dereceden bir fonksiyon minimizasyonu problemi ortaya çıkmaktadır. Ceza fonksiyonunun türevinin alınamaması nedeniyle (non-differentiable) direkt olarak, düşüm (gradient) tipi metodlar uygulanamaz. Kırık noktaların yarattığı bu zorlukların üstesinden gelmek için, doğrusal olmayan geliştirilmiş (modified) bir programlama tekniği kullanılmıştır. Sonuç olarak, bu araştırmada, sözkonusu kırıklı ceza fonksiyonunun minimizasyonu için yeni bir algoritma sunuyoruz.

Bu algoritmada, negatif olmama (non-negativity) kısıtlayıcıları düşürülerek ve " Conjugate Gradient Method " 'u kullanılarak, maximum eşlenik yönler kümesi hesaplanır. Daha sonra, ceza fonksiyonunu minimizasyonu için, bu yönler üzerinde sırasal doğru taramaları yapılır. Optimal ölçüt sağlanamadığı ve bütün yönlerdeki ilerlemelerin (improvements) yeterli olmadığı durumda, " Conjugate Gram-Schmit Süreç " 'i ile, yeni eşlenik yönler kümesi hesaplanır. Fakat, bu yönlerin birisi, bulunan noktadaki alt türevselinin (subdifferential) elemanıdır.

Anahtar Kelimeler: Doğrusal olmayan progralama, Doğrusal programlama, Uyumluluk problemi, Ceza fonksiyonu.

To my wife and my parents,

ACKNOWLEDGEMENT

I would like to thank to Assoc. Prof. Osman Oğuz for his supervision, guidance, suggestions, and encouragement throughout the development of this thesis. I am grateful to Prof. Halim Doğrusöz, Assoc. Prof. Mustafa Akgül, Prof. Akif Eyler and Assoc. Prof. Péter Kas for their valuable comments.

I would like to extend my deepest gratitude and thanks to my wife for her morale support, encouragement, especially at times of despair and hardship.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In this chapter we summarize some of the existing theory and algorithms for solving the systems of linear inequalities and then we go over some methods for constrained optimization by using a non-differential penalty function and finally we give briefly the conjugate subgradient algorithms for non-smooth optimization.

1.1 Relaxation Methods for Solving Systems of Linear Inequalities

Agmon [1] presented the method of relaxation to solve a system of linear inequalities in which at each iteration one inequality that is violated, is choosen in some way (the inequality which has the maximum distance from the current iterate or the inequality with the maximum residual or the inequality which is chosen by a prearranged cyclical sequence). The method continues by moving in the direction of the inner normal to the chosen halfspace (inequality) by a multiple λ of the distance from the current point to that halfspace. In summary at k^{th} iteration one starts from point x^k and if $\lambda_k = 1$ then x^{k+1} is the orthogonal projection of x^k on the chosen halfspace and if $\lambda_k = 2$ then x^{k+1} is the reflection of x^k. We may have overprojection, that is $\lambda_k \in (1,2)$ or underprojection $\lambda_k \in (0,1)$.

The convergence proof was done by Agmon, Motzkin and Schoenberg [14] based on the fact that, under any of the three implementations of the relaxation method as applied to a nonempty polyhedron P (P is the feasible region of the system) and for $\lambda \in [0,2]$ the relaxation sequence $\{x^k\}$ satisfies:

$$\|x^{k+1} - z\| \leq \|x^k - z\| \qquad \forall z \in P$$

which implies the Fejer-Monotone sequence of the points $\{x^k\}$ with respect to P. Fejer showed that any Fejer-Monotone sequence is convergent.

Using some lemmata basic to Khachian's polynomially bounded algorithm Telgen [19] showed that the relaxation method is finite in all cases and so can handle infeasible problems as well. However the worst case behavior of the relaxation method is not polynomially bounded and a class of problems are constructed on which the relaxation method requires an exponential number of iterations.

The basic advantages of the relaxation methods is that, these methods can be efficiently implemented to solve huge systems of linear inequalities, because there is no need for matrix inversions which causes the computational simplicity of these methods.

1.2 Surrogate Constraint Methods

Yang and Murty [23] developed a new iterative method based on the generation of a surrogate constraint from the violated inequalities in each step. The basic idea is due to fact that when solving a huge system of linear inequalities, considering only one constraint at any iteration would lead to slow convergence. They presented the following methods:

The Basic Surrogate Constraint Method: In this method at each iteration a surrogate constraint is derived by taking a nonnegative combination of all violated constraints. At the k^{th} iteration the next point x^{k+1} is on the line segment, joining x^k to its reflection with respect to surrogate constraint.

The Sequential Surrogate Constraint Method: Initially the system of linear inequalities $Ax \leq b$ is partitioned into ℓ subproblems as: $A^i x \leq b^i \quad , \quad i = 1, 2, \ldots, \ell$ Starting at point x^k the surrogate constraint is constructed from the set of violated constraints of the first subproblem and the basic method is followed for obtaining x^{k+1}. In the next iteration the surrogate constraint is constructed from second subproblem but now the starting point is x^{k+1} and following the basic method we reach to x^{k+2} and so on. In this way the algorithm goes through major cycles, in every major cycle, each of the ℓ subproblems is operated on once in serial order $\quad i = 1, 2, \ldots, \ell$.

Parallel Surrogate Constraint Method: This method is similar to second one but now the surrogate constraint is constructed by considering all of the subsystems $A^i x \leq b^i \quad , \quad i = 1, 2, \ldots, \ell$ simultaneously, that is if we are at point x^k then this point is a starting point for all subsystems. Say the resulting new points based on basic method be $x_{(i)}^{k+1} \quad i = 1, 2, \ldots, \ell$ then x^{k+1} is the convex combination of the points $x_{(i)}^{k+1} \quad i = 1, 2, \ldots, \ell$.

1.3 Khachiyan's Algorithm

Khachiyan [18] showed that the ellipsoid method can be modified in order to check the feasibility of a system of linear inequalities in polynomial time. For finding a feasible point of the system of $Ax \leq b$ (where $A \in \mathcal{R}^{m+n}, x \in \mathcal{R}^n, b \in \mathcal{R}^m$) the algorithm starts with the initial ellipsoid $E_0 = E(A_0, x_0)$ where $x_0 = 0, A_0 = R^2 I$, and R is a real number large enough to have $P \subseteq S(0, R)$, here P is the set of feasible region. At any iteration k we have $E(A_k, x_k)$ containing P, if $x_k \in P$ then the algorithm stops, but if $x_k \notin P$, then the most violated inequality is chosen, say $a_\alpha x \leq b_\alpha$ and for the feasible part of the ellipsoid which is cut off by this constraint, a new ellipsoid is constructed, that is:

$$E(A_{k+1}, x_{k+1}) = E(A_k, x_k) \cap \{x : \ a_\alpha x \leq a_\alpha x_k\}.$$

Hence at each iteration the volume of ellipsoid is decreased by a factor $q_n < e^{-\frac{1}{2n}} < 1$.

The problem is infeasible if $k = N$, that is the volume of the ellipsoid V_f is small enough to declare P is empty. Where V_0 is the volume of E_0 and

$$V_f = (e^{-\frac{1}{2n}})^{N'} V_0$$

$$N = \lceil N' \rceil \quad \text{that is the smallest integer greater or equal to } N'.$$

N is an upper bound for the number of iterations, and we can set:

$$V_f \approx 2^{-(n+1)<A>+n^3} \quad \text{where } < A > \text{ is the encoding length of } A.$$

and

$$N = 2n\left((2n+1) < A > +n < b > -n^3\right)$$

$$R = \sqrt{n}\, 2^{<A,b>-n^2} \quad \text{where } < A, b > \text{ is the encoding length of } A \text{ and } b.$$

1.4 CONN's Projection Method

Consider the problem of finding the constrained minimum x^* of the function f on the set

$$F = \{x \in R^n \mid \quad \phi_i(x) \geq 0, \quad i = 1, 2, \ldots, k\} \tag{1}$$

where $\mathcal{R}^n$ is an n-dimensional real vector space, $f, \phi_i (i = 1, 2, \ldots, k)$ are real continuous functions on $\mathcal{R}^n$.

One method of solution is to minimize the well known penalty function [6],[7]

$$P_0(x) = \mu f(x) - \sum_{i=1}^{k} min(0, \phi_i(x)) \tag{2}$$

for $x \in \mathcal{R}^n$, $\mu \geq 0$. Given μ let $x(\mu)$ be the minimum of this function. It is known that for sufficiently small μ, $x(\mu) = x^*$. Conn[6] presented a method that enables a modified form of the gradient type approach to be applied to a perturbation of the penalty function P_0 above.

Define $I(x,\varepsilon) = \{i \in k : \mid \phi_i(x) \mid > \varepsilon\}$ and $A(x,\varepsilon) = K \backslash I(x,\varepsilon)$. Here $I(x,\varepsilon)$ is the index set of those ϕ_i that may be considered inactive in a neighborhood of x, and $A(x,\varepsilon)$ be the active constraints at x. Starting with x^i and suitable choice of μ and ε (a small positive number), the algorithm separates the set of ε active and inactive constraints. Dropping the active constraints from P_0 enables us to find the gradient of P_0 at x^i. Projection of $-\nabla P_0$ onto nullspace of gradient of active constraints gives the search direction, d_i. By line search we find x^{i+1} and then redetermine the set of active and inactive constraints and continue the iterations.

Later Conn and Pietrzykowski [8] presented a method for the same penalty function that constructs a single sequence which minimizes P_0 and converges directly to x^*. This method is the continuation of their earlier results. For the case of $\varepsilon = 0$ we may exhibit the zigzagging when ϕ_i 's are nonlinear. According to this reason it is necessary to consider projecting when the constraints are not necessarily exactly zero. So the stipulation of near-zero (ε active) was necessary to avoid zigzagging in the nonlinear case. However, since in general this tolerance (ε) is rather small, it becomes desirable, specially in the initial steps, to consider some larger value, i.e., ε . But this form of projection made it difficult to satisfy the constraints exactly, a phenomena which is clearly most detrimental as we approach final convergence. Their new method overcomes this difficulty as follows: At each iteration the direction of search has two components, the first which is called horizontal component, is Conn's projected gradient that we mentioned before. The second direction or the vertical component makes a linearization to satisfy the relevant constraints exactly.

1.4.1 Conjugate Subgradient Methods and Extension of Them to a Class of Bundle Methods

Wolfe [20] and Lemarechal [13] in their methods for Non-Smooth Optimization (N.S.O) use a bundle method, in which the direction of search at any iteration is computed from a set of subgradients which have been found so far. In our algorithm we will use a bundle of conjugate directions and since there are some similarities in this sence between our method and Wolfe's conjugate subgradient algorithm, we give a summary of the theory of the methods used in [13],[20] . Lemarechal [13] showed that the methods of conjugate gradients are perfectly justified whenever a local aspect is concerned, but

that this local aspect is not enough for constructing efficient algorithms. So he replaces the local concept by finite neighborhood and defines the class of bundle methods. Now we give a brief description of local aspect, finite neighborhood aspect and bundle method as follows:

Throughout this section $f(x)$ is a convex and Lipschitz function defined on $\mathcal{R}^n$; as a Lipschitz function f has a gradient almost everywhere in $\mathcal{R}^n$ and $\mathcal{R}^n$ is considered as a Hilbert space.

Local aspect: In the study of local aspect [13], the basic point is that we fix a point x and try to find a descent direction, i.e. we want to find a direction d such that: $f'(x,d) < 0$ where:

$$f'(x,d) = \lim_{t \to 0^+} \frac{f(x+td) - f(x)}{t}$$

Since f is Lipschitz function then it is possible to construct a sequence $\{x_i\}$ such that $\nabla f(x_i)$ exists and $x_i \to x$. By defining

$$M(x) = \{g |\ g = \lim \nabla f(x_i), x_i \to x, \nabla f(x_i) \text{ exists}\},$$

we obtain the following result [13]:

$$f'(x,d) = \sup\{< d, g > |\ g \in M(x)\}.$$

The basic idea for constructing a descent direction or, equivalently, for constructing $M(x)$ is as follows:

Assume we know k points in $M(x)$

$$G = (g_1, \ldots, g_k) \subset M(x)$$

This can be initialized by computing $g_1 = \nabla f(x)$. Now we show that we can find either a descent direction or determine some $g_{k+1} \in M(x)$ so as to improve the current approximation $M(x)$ of $\nabla f(x)$. Since $f'(x,d) \geq \max\{< d, g_i > |\ i = 1, \ldots, k\}$ we choose some d_k (The set of descent directions is the (open) polar cone of the convex cone generated by $M(x)$) such that

$$< d_k, g_i >< 0, \qquad i = 1, \ldots, k. \tag{3}$$

Now two cases may occur:

i) either there exists $t > 0$ such that $f(x + td_k) < f(x)$ then d_k is a descent direction, and we are done,

ii) or $f(x + td_k) \geq f(x)$. $\qquad \forall t.$

In this case for any $g \in M(x + td_k)$ by using the subgradient inequality for f and for any $t > 0$ we have:

$$f(x + td_k) \geq f(x) \geq f(x + td_k) + < g, x - x - td_k > = f(x + td_k) + < g, -td_k > .$$

Let $t \to 0^+$ and denote by g_{k+1} any cluster point of g , then $g_{k+1} \in M(x)$ by definition. Furthermore, since $< g_{k+1}, d_k > \geq 0$, then increasing k by 1 and computing new direction satisfying (3) guaranties to obtain a new direction. In order for g_{k+1} to be as good as possible, d_k is found by solving:

$$\min_d \max\{< d, g_i > | \quad i = 1, \ldots, k\}.$$

and we find d_k as:

$$d_k = -Nr \quad \{g_1, \ldots, g_k\},$$

where $Nr\ S$ is the unique point in the closure of convex hull of the set S with minimal norm.

Finite neighborhood aspect:[13] There are some cases that $M(x)$ is either singleton or is too small and contains only limited number of subgradients. In this cases $M(x)$ may be useless or the algorithm for minimizing f by the descent directions that are found by the above procedure, may be slow.

For these cases Lemarechal gives another procedure in which the concept of $x_i \to x$ is substituted by a finite neighborhood, $v_\varepsilon(x)$ for a small $\varepsilon > 0$ which means: x_i close enough to x.

Defining $M_\varepsilon(x)$ to be:

$$M(x) \subset M_\varepsilon(x) = \{g \mid \quad g = \lim \nabla f(x_i), \quad x_i \to y, \quad y \in v_\varepsilon(x)\}.$$

by $v_\varepsilon(x)$ we mean the ε neighborhood of x .(i.e. a ball)

Its construction is easy and it is never singleton (if f is not linear). Assume d is such that $< d, g > < 0 \quad \forall g \in M_\varepsilon(x)$ (we choose such a d in the (open) polar cone of the convex cone generated by $M_\varepsilon(x)$) then $f(x + td)$ is a decreasing function of t as long as $x + td \in v_\varepsilon(x)$. Since $\varepsilon > 0$ and any line-search will get us out of $v_\varepsilon(x_i)$ then an algorithm based on this process of search will be finite. Of course like the " local aspect " study if $f(x + td) \geq f(x) \quad \forall t$, then in the same way it can be shown that we can find some $g_{k+1} \in M_\varepsilon(x)$ to improve the current approximation $M_\varepsilon(x)$ of $\nabla f(x)$.

Bundle Methods:[13] In this part we assume that we have performed the descent algorithm discussed above for the points $x_1, \ldots, x_k$ and for any point x_i we have one subgradient g_i and the value of f at x_i, f_i.

So symbolize these information by the bundle:

$$x_1, \ldots, x_k; \qquad f_1, \ldots, f_k; \qquad g_1, \ldots, g_k$$

Where $f_i = f(x_i)$, $g_i \in \partial f(x_i)$ and $\partial f(x)$ is the set of all subgradients of f at x. Say $G_k = \{g_1, \ldots, g_k\}$.

In some situations we are unable to compute exactly the function and it's gradient at x_i, then for any x_i together with some prescribed tolerance ε_i we find f_i and g_i such that:

$$f(x_i) - \varepsilon_i \leq f_i \leq f(x_i) \quad \text{and } g_i \in \partial_{\varepsilon_i} f(x_i)$$

We use the above bundle to compute x_{k+1} and g_{k+1} as follows:

Let $G_k \subset \partial f(x_k)$ and choose $d_k = -Nr\ G_k$, do line search and find $x_{k+1} = x + t_k d_k$, where t_k is the step length and then compute $g_{k+1} \in \partial f(x_{k+1} + t_k d_k)$. There are two cases to be considered:

1) When all the points from x_1 to x_k are close together and all the g_i's are approximately in $\partial f(x_k)$.

2) If f is quadratic and we do exact line-searches from x_1 to x_k then in this case $Nr\ G_k$ turns out to be the direction of conjugate gradients.

It must be noticed that for k being large the direction $d_k = -Nr\ G_k$ is a poorly justified choice. To solve this difficulty we need to use a reset step by deletting those g_i's that appear to be poor approximation of $\partial f(x_k)$. In this way k is forced to be small and this gives variants of conjugate subgradient methods.

1.5 Wolfe's Conjugate Subgradient Method

Let the polyhedral, convex function f has the form :

$$f(x) = \max\{f_i(x), \quad i = 1, 2, \ldots, m\},$$

where the function f_i are affine, so that $\nabla f_i(x) = g_i$ is constant for each i ; f and $f'(x; \cdot)$ are closed and proper. Define

$$I(x) = \{i : \ f_i(x) = f(x)\}.$$

for all x, then

$$\partial f(x) = conv\{g_i : \ i \in I(x)\}.$$

For any set $S \subseteq E^n$, there is a unique point v in the closure of convex hull of S having minimal norm, we denote it by $Nr\ S$. i.e. if $Nr\ S = v$ then:

$$< v, s > \geq \|v\|^2 \qquad \text{for all} \quad s \in S.$$

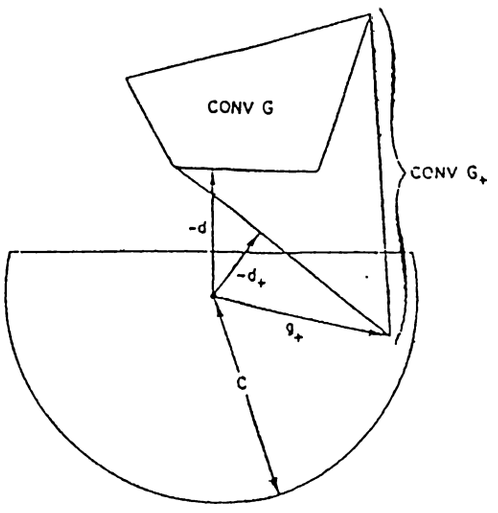


Figure 1.1: Constructing G_+, d_+ in Wolfe's Method.

By using the following characterization of gradient:

$$\nabla f(x) = -Nr \ \ \partial f(x),$$

We can write $\nabla f(x) = -Nr \ \ (conv\{g_i : \ i \in I(x)\}) = -Nr \ \ \{g_i : \ i \in I(x)\}$.

Now assuming that we have a bundle $G_k = \{g_{k-p}, \ldots, g_{k-1}, g_k\}$, (where p is a natural number less than k) we set $d_k = -Nr \ \ G_k$ be the direction of steepest descent of f at x_k.

Wolfe [20] presented a conjugate subgradient method for minimizing non differentiable functions, which is based on bundle methods with a special line search. Assuming that we have a bundle

$$x_0, \ldots, x_k \quad , \quad g_0, \ldots, g_k \quad , \quad f(x_0), \ldots, f(x_k)$$

the algorithm works as:

i) Compute $d_k = -Nr \ \ G_k \quad$ where $G_k = \{g_0, \ldots, g_k\}$.

ii) Line search gives $t \geq 0$, and $g_{k+1} \in \partial f(x + td_k)$.

iii) Set $G_{k+1} = \{G_k, g_{k+1}\}$.

iv) If the optimality conditions are not satisfied, go to (i), else stop.

Computing G_+, d_+ $(G_+ = G_{k+1}, \ d = d_k, \ d_+ = d_{k+1})$ is shown geometrically in Figure 1.1.

If $x_0, \ldots, x_k$ are close together, the above iteration continues until $\|d_k\| < \varepsilon$ for some $k = K$ and then if

$$\sum_{k<K} \|x_{k+1} - x_k\| \leq \delta$$

is satisfied, the optimality conditions can be checked, if not, a reset step is begun.

Resetting is done by choosing $G_k = \{g_k\}$, that is by discarding all the previous subgradients. Wolfe's algorithm constitutes an extension of the method of conjugate gradients method of Hestenes and Stiefel [11].

2. A NEW CONJUGATE GRADIENT APPROACH FOR NON-SMOOTH OPTIMIZATION

We consider the feasibility problem :

$$\mathbf{P}_0 \qquad s.t. \quad \bar{A}\bar{x} \leq \bar{b}$$
$$\bar{x} \geq 0$$

where $\bar{A} \in \mathcal{R}^{m \times n}$, $\bar{x} \in \mathcal{R}^n$, $b \in \mathcal{R}^m$; and $\bar{A}$ is not necessarily full raw rank. Problem $\mathbf{P}_0$ is equivalent to:

$$s.t. \quad \hat{A}x = \bar{b}$$
$$x \geq 0$$

where $\hat{A} = \left[\bar{A} \mid I_m\right] \in \mathcal{R}^{m \times (m+n)}$, $x \in \mathcal{R}^{m+n}$.

by adding the slack variables and identity I_n to $\bar{A}$ we have nonsingular matrix Q as:

$$Q = \begin{pmatrix} \bar{a}_{11} & & & \bar{a}_{1m} & 1 & 0 & & 0 \\ \vdots & & & \vdots & 0 & & & \vdots \\ \vdots & & & \vdots & \vdots & & & 0 \\ \bar{a}_{m1} & \cdots & \cdots & \bar{a}_{mn} & 0 & \cdots & 0 & 1 \\ 1 & 0 & & 0 & 0 & \cdots & & 0 \\ 0 & & & \vdots & \vdots & & & \vdots \\ \vdots & & & 0 & \vdots & & & \vdots \\ 0 & & 0 & 1 & 0 & \cdots & \cdots & 0 \end{pmatrix}$$

Then $A = Q^T Q$ is a positive definite matrix and we make the transformation:

$$b = Q^T b^*$$

Where we define b^* as:

$$b^* = \begin{bmatrix} \bar{b}_1 \\ \vdots \\ \bar{b}_m \\ \bar{b}_1^* \\ \vdots \\ \bar{b}_n^* \end{bmatrix}$$

and where $\bar{b}_i^* \quad , i = 1, 2, \ldots, n$ are choosen as:

$$\bar{b}_i^* = \frac{\sum_{j=1}^{m} \bar{b}_j}{m} \quad \forall i = 1, 2, \ldots, n.$$

It is known that if we drop the non-negativity constraints, then it is possible to solve the system of $Ax = b$ by conjugate gradient method which terminates in k iterations such that $k \leq n + m$ (here $A \in \mathcal{R}^{(n+m)\times(n+m)}$). By conjugate directions we mean A-conjugate or A-orthogonal directions, so:

$$d_i A d_j = 0 \quad i \neq j \ \ \forall i, j$$

$$d_i A d_i \neq 0 \quad \forall i$$

In fact since A is positive definite, then: $d_i A d_i > 0 \ \ \forall i$. As mentioned in section 2.5 Wolfe's algorithm constitutes an extension of the method of conjugate gradients method of Hestenes and Stiefel. Based on this we will introduce a set of A-conjugate directions $\{d_i\}_{i=1}^{n+m}$ and perform search on these directions successively. Whenever it is necessary we will compute the conjugate directions by Gram Schmit orthogonalization process.

For the feasibility problem P_0 we define the penalty function $F_1(x)$ as follows:

$$F_1(x) = \sum_{i=1}^{m} (\hat{a}_i x - \bar{b}_i)^2 + \sum_{j=1}^{n+m} [\min(0, x_j)]^2$$

Recalling that $A = Q^T Q$ and $b = Q^T b^*$, we drop the non-negativity constraints and solve the system of $Ax = b$. Assume that we have found the conjugate directions $\{d_i\}, i = 1, \ldots, n + m$ and let the step sizes in any direction d_i, be α_i, $i = 1, \ldots, n + m$. then we can write the solution as:

$$x^* = x^0 + \alpha_1 d_1 + \ldots + \alpha_{n+m} d_{n+m}$$

where x^0 is any arbitrary starting point. Solving the system of $Ax = b$ is equivalent to minimizing the:

$$\hat{F}(x) = [Ax - b]^t [Ax - b]$$

Starting from x^0 then the step size in direction d_k would be α_k

$$\hat{F}(x^0) = [Ax^0 - b]^T[Ax^0 - b]$$

$$\hat{F}(x^1) = [A(x^0 + \alpha_k d_k) - b]^T[A(x^0 + \alpha_k d_k) - b]$$

$$\frac{\partial \hat{F}(x^1)}{\partial \alpha_k} = 0 \Longrightarrow 2(Ad_k)^T[A(x^0 + \alpha_k Ad_k) - b] = 0$$

$$(Ad_k)^T Ax^0 + \alpha_k (Ad_k)^T (Ad_k) - (Ad_k)^T b = 0$$

$$\alpha_k = \frac{(Ad_k)^T b - (Ad_k)^T Ax^0}{|Ad_k|_2^2}$$

$$\alpha_k = \frac{(d_k)^T A^T b - (d_k)^T A^T Ax^0}{(d_k)^T A^T Ad_k}$$

Here α_k is the optimal step length in direction d_k without considering the non-negativity constraints and let $\alpha = \sum_{i=1}^{n+m} |\alpha_i|$.
Assume we have $\bar{b}_i^*, i = 1, 2, \ldots, n$ such that the unique solution of $Qx = b^*$ is a feasible solution to $\mathbf{P_0}$ and let the set of A-conjugate $\{d_i\}_{i=1}^{n+m}$ are given, if we use the optimal step sizes $\{\alpha_i\}_{i=1}^{n+m}$, we have:

$$\sum_{i=1}^{m} \left[\hat{a}_i \left(x^0 + \sum_{j=1}^{n+m} \alpha_j d_j \right) - \bar{b}_i \right]^2 = 0$$

since $x = x^0 + \sum_{j=1}^{n+m} \alpha_j d_j$ is a solution.

Now consider using step sizes $\bar{\alpha}_i, i = 1, 2, \ldots, (n+m)$ rather than $\alpha_i, i = 1, 2, \ldots, (n+m)$ and we choose θ as:

$$\theta = \min_j \left\{ \left| \frac{\bar{\alpha}_j}{\alpha_j} \right| \ : \ \alpha_j \neq 0 \right\}$$

then we have the relative reduction:

$$\frac{\sum_{i=1}^{n+m} (q_i x^0 - b_i^*)^2 - \sum_{i=1}^{n+m} \left[q_i \left(x^0 + \theta \sum_{j=1}^{n+m} \alpha_j d_j \right) - b_i^* \right]^2}{\sum_{i=1}^{n+m} (q_i x^0 - b_i^*)^2} = 2\theta - \theta^2$$

That means if we define error at x^k as

$$\Delta^k = \sum_{i=1}^{n+m} \left(q_i x^k - b_i^* \right)^2,$$

then we have

$$\Delta^{k+1} = \Delta^k (1 - 2\theta + \theta^2)$$

Since $\Delta \leq 2^L$ (here L is the size of the system of linear equalities) then we can stop after β iterations for which

$$2^L \left(1 - 2\theta + \theta^2\right)^\beta \leq \frac{1}{2^L}$$

Which means we can bound the number of iterations as

$$\beta \leq \frac{-2L \log 2}{\log (1 - 2\theta + \theta^2)}$$

This definitely implies a polynomial bound for for problem $\mathbf{P}_0$ for any $1 > \theta > 0$. In this work we could not prove the existence of such θ , but the computations results in the next chapter seems to support this idea that practically our algorithm can solve problems in polynomial time bound. It means that if in any loop the improvement in any direction is not posssible, we can remove that direction from the set of conjugate directions and the procedure reduces to search in the directions remained in the bundle of directions, and then it is possible to find a $0 < \theta < 1$, such that we can improve in all remaining directions.

The Main Algorithm:

i) Choose x^0 in $\mathcal{R}^{n+m}$, the starting point, and sufficiently small $\varepsilon, \delta \geq 0$, $\lambda = 0$.

ii) Compute the conjugate gradient directions $\{d_i\}, i = 1, \ldots, n+m$ by the method of C.G. algorithm and normalize them.

iii) **For** $i = 1$ to $n+m$ do

$$d_\ell = d_i, \text{ do line search, } x^+ = x + td_\ell$$

$$\lambda = \lambda + |t|$$

if $F_1(x^+) < \varepsilon$ then stop, a feasible solution is found.
else check the optimality test, if x^+ is optimal, then stop.
end (For).
if $\min\left(\frac{\lambda}{\alpha}, \lambda\right) < \delta$ then (Reset) go to step (iv).
else set $\lambda = 0$ and repeat step (iii).

iv) Set $v_1 = -\nabla F_1(x^+)$, choose $v_2, \ldots, v_{n+m}$ such that $v_1, v_2, \ldots, v_{n+m}$ are independent.

v) Find conjugate directions $\{d_i\}, i = 1, 2, \ldots, n+m$ by the procedure of C.G.Schmit, normalize them, set $\lambda = 0$ and go to step (iii).

Convergence Results:

Definition: Let W be the set of minimum points of $F_1(x)$ and let $W \neq \emptyset$. A sequence $\{x^k\}_{k=1}^{\infty}$ in $\mathcal{R}^n$ is called strictly Fejer-Monotone with respect to the set W, if for every $x \in W$ we have:

$$\|x^{k+1} - x\| < \|x^k - x\| \text{ for all } k \geq 1$$

Every Fejer-Monotone sequence is bounded if $W \neq \emptyset$, since $\|x^k - x\|$ is always positive and monotonically decreasing with W [23].

Definition: Let $\{d_i\}_{i=1}^{n+m}$ be A-conjugate directions such that $d_1 = -g_f(x^k)$ and $d_2, d_3, \ldots, d_{n+m}$ are generated by C.G. or C.G.S. algorithms with respect to d_1 and a positive definite matrix A. We say a loop k is processed if starting from point x^k, a set of line searches are done on all directions of the set $\{d_i\}_{i=1}^{n+m}$ and we reach the new point x^{k+1}. Of course this process is equivalent to $n+m$ iterations.

Theorem 1 (basic idea due to Shor[17]) *let F be a convex function and $W \neq \emptyset$, then any sequence $\{x^k\}_{k=1}^{\infty}$ generated by the algorithm above is strictly Fejer-Monotone*

with respect to W *and for any* $\varepsilon > 0$ *and any* $x^* \in W$ *there exists a* $\bar{k}$ *and* $\bar{x}$ *such that* $F(\bar{x}) = F(x^{\bar{k}})$ *and:*

$$\|\bar{x} - x^*\| < \frac{h}{2}(1+\varepsilon)$$

Where h *is a real positive number.*

Proof: Without loss of generality assume that in any k^{th} loop the directions $\{d_i\}_{i=2}^{n+m}$ are not suitable for improvement in line searchs. Then our algorithm reduces to a subgradient type method, that in any loop we have:

$$x^{k+1} = x^k - h_{k+1}\frac{g_f(x^k)}{\|g_f(x^k)\|}$$

$$\text{where:} \begin{cases} h_{k+1} & = \text{steplength} \\ g_f(x^k) & = g^k = \nabla F(x^k) \end{cases}$$

$$\begin{aligned} \text{let } x^* \in W \Rightarrow \|x^{k+1} - x^*\|^2 &= \|x^k - x^* - h_{k+1}\tfrac{g^k}{\|g^k\|}\|^2 \\ &= \|x^k - x^*\|^2 + h_{k+1}^2 - 2h_{k+1} < x^k - x^*, \tfrac{g^k}{\|g^k\|} > \end{aligned}$$

if $g^k = 0$ then $\bar{x} = x^k = x^*$.

So let $g^k \neq 0$ and take $\ell_k =< x^k - x^*, \frac{g^k}{\|g^k\|} >$ which is distance from x^* to the hyperplane:

$$H_k = \{x : < g^k\ ,\ x^k - x >= 0\}$$

Define

$$D_k = \{x :\ F(x) \doteq F(x^k)\}$$

$$b_k(x^*) = \min_{x \in D_k} \|x^* - x\|$$

Since D_k and x^* are in one side of H_k , then any segment joining the point x^* to a point of H_k passes through D_k. so we have :

$$\ell_k(x^*) \geq b_k(x^*)$$

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 + h_{k+1}^2 - 2h_{k+1}b_k(x^*)$$

let h be sufficiently small positive number as a constant stepsize:

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 + h^2 - 2hb_k(x^*)$$

now if $b_k(x^*) \leq \frac{h}{2}$ for all $k = 1, 2 \ldots$

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 - \varepsilon h^2 \leq \|x^k - x^*\|^2 - \varepsilon(1+k)h^2$$

But since $\|x^{k+1} - x*\| \geq 0$ then there exists $\bar{k}$ such that:

$$b_{\bar{k}}(x^*) = \min_{\bar{x} \in D_{\bar{k}}} \|x^* - \bar{x}\| < \frac{h}{2}(1+\varepsilon)$$

so no matter how small ε or h is, we can find a $\bar{k}$ such that $x^{\bar{k}}$ is the minimizer of $F(x)$ and

$$\|\bar{x} - x^*\| < \frac{h}{2}(1+\varepsilon)$$

Optimality Conditions:

Ben-Tal and Zowe [2] derived the necessary and sufficient optimality conditions for the exterior penalty function:

$$\mathbf{P}_e \qquad \min f(x) = h_0(x) + \frac{1}{2}\rho \sum_{i=1}^{s} [\max\{0, h_i(x)\}]^2.$$

First, it is convenient to define the following index sets:

$$\begin{aligned} E = &\ \{i|\ \ h_i(x^*) = 0\} \\ E^+ = &\ \{i|\ \ h_i(x^*) > 0\} \end{aligned}$$

Theorem 2 *Necessary and sufficient, optimality conditions for problem* $\mathbf{P}_e$

(a) A necessary condition for x^* *to be a local minimizer of problem* $\mathbf{P}_e$ *is that*

$$h_0'(x^*) + \rho \sum_{i \in E^+} h_i(x^*) h_i'(x^*) = 0$$

and for every $d \in \mathcal{R}^{n+m}$,

$$h_0''(x) < d, d > + \rho \sum_{i \in E^+} h_i(x^*) h_i''(x^*) < d, d > + \rho \sum_{i \in E} [\max\{0, h_i'(x^*) d\}]^2$$
$$+ \rho \sum_{i \in E^+} [h_i'(x^*) d]^2 \geq 0.$$

(b) A sufficient condition for x^* *to be an isolated local minimizer for problem* $\mathbf{P}_e$ *is as above but with strict inequality for* $d \neq 0$.

For our penalty function we have:

$$\begin{aligned} \rho = &\ 2 \\ s = &\ n + m \\ h_i(x) = &\ -x_i (\text{the negative of the } i^{th} \text{ element of } x) \end{aligned}$$

and

$$h_0(x) = \sum_{i=1}^{m} (\hat{a}_i x - \bar{b}_i)^2$$

Optimality Test: We have a set of conjugate directions $D = \{d_i\}_{i=1}^{m+n}$, then a point x^* is an optimal solution of $F_1(x)$ if sufficiency condition of optimality is satisfied for any d_k and $-d_k$ such that $d_k \in D$, $k = 1, 2, \ldots, n+m$. Since any set of conjugate directions are also independent, then any arbitrary direction d_e can be written as the linear combination of the elements of the set D , that is $d_e = \sum_{k=1}^{m+n} r_k d_k$, $r_k \in \mathcal{R}$ and $d_k \in D$.

So if sufficient conditions for optimality at a point x^* is satisfied for any d_k and $-d_k$ such that $d_k \in D$, $k = 1, 2, \ldots, n+m$, then it will be satisfied for any $d_e \in \mathcal{R}^{n+m}$ and in this way the existing point x^* will be optimal solution.

Of course the optimality test is not necessary for feasible problems and for the attempt feasibility problems we never used this test (for feasible problems), because $\min F_1(x^*) = 0$ for a feasible point x^*. But in application of our algorithm for **LP** problems we need to use this optimality test.

3. NUMERICAL EXPERIMENTATION

We decided that for our A-conjugate method, we need to check the effectiveness of the algorithm practically. So we generated a series of random problems in different sizes and applied our algorithm to solve them. In these problems, that we give in tabular form in this chapter, the size of A changes from $\mathcal{R}^{10\times10}$ to $\mathcal{R}^{40\times40}$, and for larger problems we need to modify the C.G. algorithm to find exact A-conjugate directions.

Recalling from the last chapter, although we could not guaranty the existence of $0 < \theta < 1$ in any loop of iterations (n+m iterations) to support the polynomiality motivation of our algorithm, but as it is seen from the computations results in this chapter, the method solves problems in polynomial time bound practically at least for the size range of problems given above.

Due to numerical errors the accuracy of search directions $\{d_i\}_{i=1}^{n+m}$ (generated by C.G. or C.G.S. processes) to be A-conjugate is decreased as size of A increases. The sparsity of the matrix $\bar{A}$ for all the solved problems is over 80 percent (that is the number of non-zero elements of $\bar{A}$ is more than 80 percent of its all elements) besides there is not any conditions on $\bar{A}$ and $\bar{b}$.

The important result from numerical experimentations is that, for all solved problems the convergence is so quick in first loop, that is for the problem of size $\mathcal{R}^{n\times n}$ the penalty function is reduced exponencially at first n iterations and reachs to a relatively very small penalty value with respect to its initial value and then the process achieves a rather slow convergence. The worst case of number of iterations necessary to find a minimizer of penalty function in our test, is bounded by $15n$.

Since the symmetric and positive matrix A may be ill conditioned, for example some of its eigenvalues may be too small, then in calculating A-conjugate directions the roundoff errors may be too large and this leads to the cases where $d_i^t A d_j \neq 0$ for $i \neq j$, so the set $\{d_i\}_{i=1}^{n+m}$ can not exactly represent $\mathcal{R}^{n+m}$ space.

FACTORS	$\mathcal{K}=10$	$\mathcal{K}=13$	$\mathcal{K}=16$	$\mathcal{K}=20$	$\mathcal{K}=26$	$\mathcal{K}=30$	$\mathcal{K}=36$	$\mathcal{K}=40$
m	4	5	6	8	10	12	16	16
n	6	8	10	12	16	18	20	24
u_{ij}	20	20	20	20	20	20	20	20
l_{ij}	-20	-20	-20	-20	-20	-20	-20	-20
NZ	4	5	6	6	8	8	12	12
uc_j	10	10	10	10	10	10	10	10
lc_j	-1	-1	-1	-1	-1	-1	-1	-1
ρ_c	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8

Table 3.1: Schedule of design of test problems

$\mathcal{K}$ = dimension of the space, $A \in \mathcal{R}^{\mathcal{K}\times\mathcal{K}}$.

m = number of linear inequalities.

n = number of variables without adding slack variables.

u_{ij} = upper bound for $\bar{a}_{ij}$.

l_{ij} = lower bound for $\bar{a}_{ij}$.

uc_j = upper bound for c_j.

lc_j = lower bound for c_j.

NZ = number of non-zero elements in every column $\bar{a}^j$

ρ_c = density of number of non-zero elements of c.

$\frac{\lambda}{\alpha}$ = total (relative) step length during one loop $(m+n$ iterations).

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	3	0.003860	0.017921	0.017921
2	4	0.007682	0.004227	0.004227
3	3	0.000367	0.004419	0.004419
4	3	0.007072	0.011708	0.011708
5	17	0.000874	0.003076	0.004755
6	28	0.010914	0.005772	0.012161
7	3	0.001199	0.003783	0.003783
8	3	0.001846	0.004212	0.004212
9	3	0.000119	0.001505	0.001505
10	3	0.000819	0.002757	0.002757
11	56	0.010789	0.000249	0.019070
12	19	0.006049	0.000003	0.001483
13	26	0.016264	0.003028	0.014203
14	15	0.016651	0.002718	0.007129

Table 3.2: $Ax = b\,, \quad A \in \mathcal{R}^{10\times10}$

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	5	0.000922	0.004142	0.004142
2	5	0.013952	0.002545	0.002545
3	20	0.003806	0.001378	0.002427
4	5	0.000570	0.009462	0.009462
5	30	0.019781	0.000052	0.006128
6	32	0.019084	0.000030	0.003567
7	5	0.010253	0.005145	0.005145
8	41	0.016802	0.000012	0.002100
9	72	0.019898	0.000094	0.010462
10	129	0.019927	0.001547	0.009797

Table 3.3: $Ax = b\,, \quad A \in \mathcal{R}^{13\times13}$

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	23	0.002494	0.000200	0.001460
2	6	0.001620	0.011088	0.011088
3	58	0.017985	0.000874	0.010186
4	54	0.009547	0.000021	0.021087
5	39	0.018295	0.000102	0.002721
6	6	0.000240	0.015218	0.015218
7	72	0.018383	0.000108	0.007236
8	6	0.000549	0.004565	0.004565
9	7	0.002867	0.009162	0.009162
10	62	0.014058	0.000364	0.010775
11	59	0.014058	0.000221	0.005854
12	87	0.015753	0.000100	0.008568
13	5	0.008931	0.002479	0.002479
14	6	0.000389	0.007353	0.007353
15	16	0.016699	0.037585	0.037585
16	6	0.000414	0.007798	0.007798
17	54	0.014726	0.000047	0.018080
18	77	0.019968	0.000305	0.009002
19	72	0.015961	0.000092	0.007809
20	151	0.019186	0.000248	0.004713

Table 3.4: $Ax = b\ , \quad A \in \mathcal{R}^{16\times 16}$

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	53	0.007171	0.002620	0.006169
2	55	0.015347	0.005015	0.007598
3	7	0.007326	0.004334	0.004334
4	9	0.006160	0.013377	0.013377
5	52	0.011085	0.000143	0.026794
6	28	0.007784	0.000488	0.001557
7	25	0.019862	0.000031	0.040541
8	112	0.019612	0.000012	0.039558
9	9	0.005295	0.014534	0.014534
10	29	0.019185	0.000306	0.017977
11	10	0.018377	0.026277	0.026277
12	50	0.008287	0.000080	0.014182
13	49	0.017067	0.000080	0.010122
14	73	0.013925	0.001308	0.010502
15	69	0.019215	0.000271	0.011067
16	74	0.013281	0.000848	0.020566
17	75	0.018330	0.000522	0.008848
18	231	0.018757	0.000734	0.002817
19	76	0.009882	0.000732	0.006515
20	56	0.014150	0.005013	0.009095
21	115	0.019891	0.000747	0.006236

Table 3.5: $Ax = b\,, \quad A \in \mathcal{R}^{20\times 20}$

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	19	0.012911	0.029228	0.029228
2	191	0.029996	0.000006	0.004158
3	36	0.027740	0.000320	0.018018
4	67	0.007212	0.000330	0.005650
5	11	0.015245	0.005242	0.005242
6	144	0.029670	0.000050	0.005359
7	10	0.007677	0.006737	0.006737
8	149	0.029920	0.000382	0.100395
9	64	0.012921	0.000136	0.004630
10	9	0.010766	0.022777	0.022777
11	68	0.027688	0.000161	0.004136
12	149	0.029920	0.000382	0.100395
13	64	0.012921	0.000136	0.004630
14	9	0.010766	0.022777	0.022777
15	68	0.027688	0.000161	0.004136
16	70	0.024106	0.002148	0.007798
17	169	0.029791	0.000020	0.010201
18	173	0.026735	0.000071	0.001617
19	64	0.026865	0.000230	0.015913
20	91	0.026094	0.000119	0.004617
21	95	0.013781	0.000310	0.006535
22	89	0.027748	0.000033	0.004375
23	143	0.023141	0.001590	0.005584
24	253	0.018113	0.002862	0.068871

Table 3.6: $Ax = b\,, \quad A \in \mathcal{R}^{26\times 26}$

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	14	0.008159	0.081600	0.081600
2	106	0.028005	0.000065	0.006770
3	134	0.024513	0.000022	0.004174
4	14	0.011539	0.007543	0.007543
5	80	0.024698	0.001907	0.019771
6	103	0.016989	0.000051	0.018905
7	200	0.027437	0.002198	0.055168
8	111	0.010574	0.001337	0.007677
9	225	0.029625	0.000014	0.001667
10	102	0.026161	0.000006	0.003876
11	352	0.029524	0.000163	0.002316
12	75	0.021587	0.000295	0.012395
13	104	0.029698	0.000009	0.004808
14	104	0.029981	0.000031	0.007279

Table 3.7: $Ax = b\,, \quad A \in \mathcal{R}^{30\times 30}$

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	196	0.028532	0.000001	0.045340
2	273	0.026510	0.000483	0.014159
3	51	0.028544	0.004593	0.123871
4	17	0.026963	0.208405	0.208405
5	47	0.023672	0.001274	0.066795
6	235	0.021095	0.000081	0.180527
7	425	0.029967	0.000001	0.197356

Table 3.8: $Ax = b\,, \quad A \in \mathcal{R}^{36\times 36}$

problem #	# of iterations	$f(x)$	min $\left(\frac{\lambda}{\alpha}\right)$	average $\left(\frac{\lambda}{\alpha}\right)$
1	143	0.028990	0.000039	0.092190
2	458	0.013592	0.000677	0.132495
3	62	0.028406	0.004004	0.048863
4	185	0.029546	0.000174	0.019071
5	17	0.026838	0.014640	0.014640
6	138	0.022210	0.000061	0.038515
7	176	0.028839	0.021943	0.540074
8	386	0.028450	0.000309	0.003927
9	318	0.029407	0.000971	0.018176

Table 3.9: $Ax = b\,, \quad A \in \mathcal{R}^{40\times 40}$

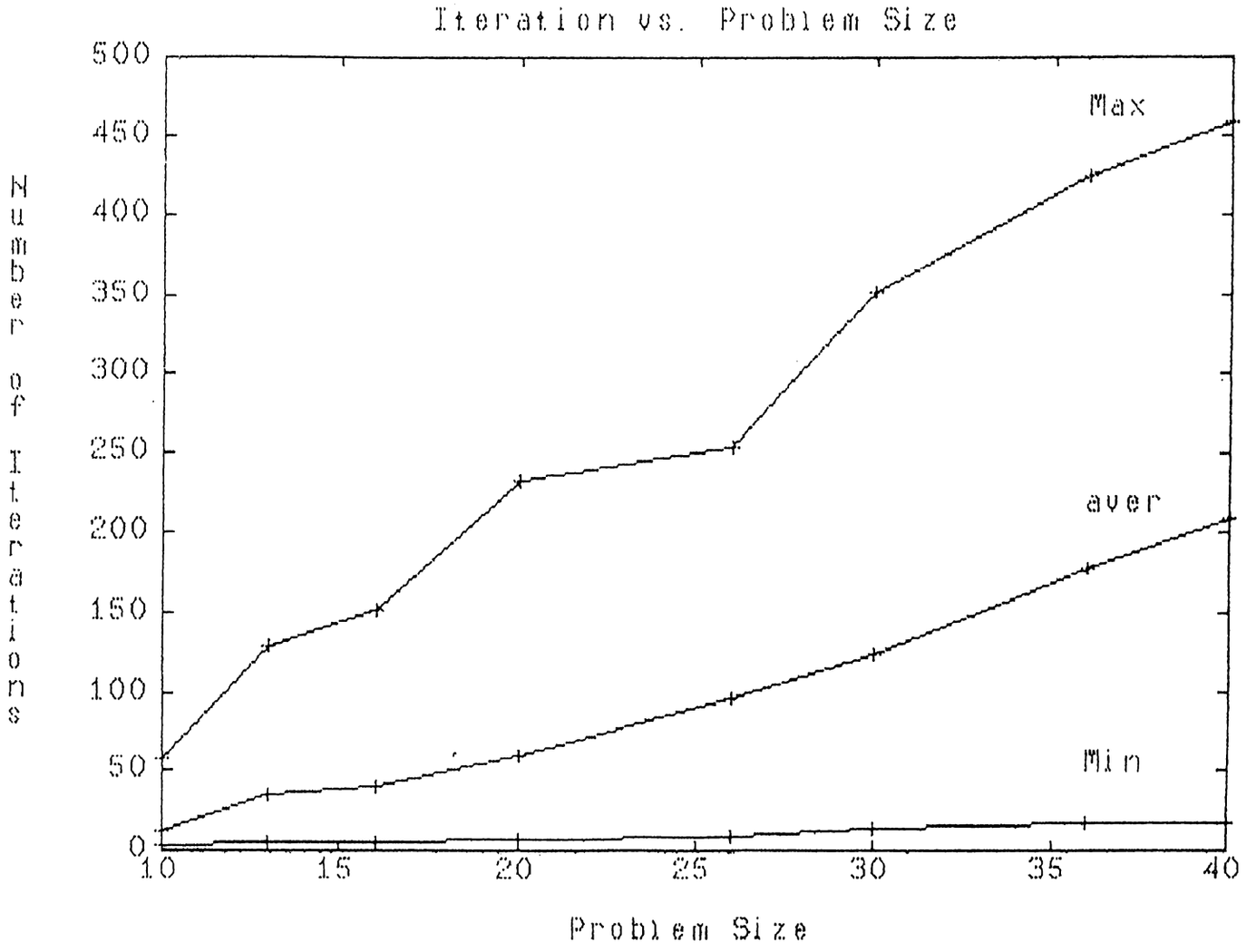


Figure 3.1: Number of Iterations vs. Problem Size

4. APPLICATIONS

4.1 Linear Inequality Models in Computerized Tomography

Computerized tomography [23] is a method in which the image of the cross section of the body is reconstructed by a computer. It can be used to show a three dimensional view of the interior structures of the human body and so CT can detect some conditions that conventional X-ray pictures can not detect. Initially the structure (which can be brain, heart, etc.)which is being studied, is divided into slices and then as finite element method any slice is considered to be divided into sufficiently small pixels. The process for a two dimensional slice is as follows:

An X-ray beam, say beam 1 penetrate the slice, entering it with the initial intensity I_1, and emerges at the detector at the end of its path through the slice, with an intensity F_1. Hence the total absorption of the energy through the path is $I_1 - F_1$. Define a_{1j} be the length of the intersetion of the path of beam number 1 with the j^{th} pixel, and say x_j be the unknown local density of the j^{th} pixel and assume the data is collected from m different beams for every slice, then we have:

$$\sum_{j=1}^{n} a_{ij}x_j = b_j \quad i = 1, 2, \ldots, n$$

$$\text{where} \quad b_j = I_j - F_j$$

Since the assumption that the local density is a constant within each pixel, is unlikely to be valid and since the local densities are not negative, so we can replace upper system by the inequality system as:

$$b_i - \varepsilon_i \leq \sum_{j=1}^{n} a_{ij}x_j \leq b_i + \varepsilon_i \qquad 1 \leq i \leq m$$

$$0 \leq x_i \leq u \qquad\qquad 1 \leq j \leq n$$

where u is a known upper bound for the density of pixels and n equals to the total number of pixels in slice. Solution of this system of linear inequalities gives densities of the object under study.

4.2 LINEAR PROGRAMMING VIA NON-SMOOTH OPTIMIZATION

In this chapter we try to extend our algorithm for solving the linear feasibility problems to solve linear programming problems. We onsider the LP problem :

$$\mathbf{P}_1 \qquad \begin{array}{ll} \max & cx \\ s.t. & \bar{A}\bar{x} \leq \bar{b} \\ & \bar{x} \geq 0 \end{array}$$

where $\bar{A} \in \mathcal{R}^{m \times n}$, c and $\bar{x} \in \mathcal{R}^n$, $\bar{b} \in \mathcal{R}^m$.

Adding slack variables we have :

$$\begin{array}{ll} \max & cx \\ s.t. & \hat{A}x = \bar{b} \\ & x \geq 0 \end{array}$$

where $\hat{A} = \left[\bar{A} \mid I_m\right] \in \mathcal{R}^{m \times (m+n)}$, and $x \in \mathcal{R}^{m+n}$.

There are alternative ways of reducing this **LP** to a feasibility problem, one way, which we have tried is the following: Recalling the exterior penalty function

$$\min f(x) = h_0(x) + \frac{1}{2}\rho \sum_{i=1}^{s} [\max\{0, h_i(x)\}]^p, \qquad p \geq 1$$

where:
- s : a natural number.
- ρ : a positive real number.
- $\mathrm{h_i(x)}$: smooth function defined on a real normed vector space.

We define the exterior penalty function $F_2(x)$ for the LP problem P_1 as:

$$F_2(x) = (cx - \bar{K})^2 + \sum_{i=1}^{m} (\hat{a}_i x - \bar{b}_i)^2 + \sum_{j=1}^{n+m} [\min(0, x_j)]^2$$

Here $\bar{K}$ is an upper bound for **cx** and we assumed that ρ (the penalty weight) equals 2 for any penalty term, and

$$h_0(x) = (cx - \bar{K})^2 + \sum_{i=1}^{m} (\hat{a}_i x - \bar{b}_i)^2$$

$$h_i(x) = -x_i \quad \text{(the negative of the } i^{th} \text{ element of } \mathbf{x} \text{)}$$

The Basic Algorithm:

i) Choose x^0 in R^{n+m}, the starting point, and $\varepsilon, \delta, \rho \geq 0, \lambda = 0$.

ii) Compute the conjugate gradient directions $\{d_i\}, i = 1, \ldots, n+m.$ by the method of conjugate gradient of Hestenes and normalize them.

iii) **For** $i = 1$ to $n+m$ do

$$d_\ell = d_i, \text{do line search}, x^+ = x + td_\ell$$

$$\lambda = \lambda + |t|$$

Check the optimality conditions, if x^+ is optimal solution, then stop.
end (For).
if $F_1(x^+) \leq \varepsilon$ then go to step(vi).
if $\min(\lambda, \frac{\lambda}{\bar{\alpha}}) < \delta$, then (Reset) go to step (iv),
else set $\lambda = 0$ and repeat step (iii).

iv) Set $v_1 = -\nabla F_1(x^+)$, choose $v_2, \ldots, v_{n+m}$ such that $v_1, v_2, \ldots, v_{n+m}$ are independent.

v) Find conjugate directions $\{d_i\}, i = 1, 2, \ldots, n+m$ by the procedure of **C.G.S.** , normalize them, set $\lambda = 0$ and go to step (iii).

vi) Improve cx by the following procedure (it needs n iterations):
consider the simplex tablaue:

$$\begin{pmatrix} c_1 & c_2 & & c_n & 0 & \ldots & 0 \\ \bar{a}_{11} & \bar{a}_{12} & & \bar{a}_{1n} & 1 & & \\ \vdots & \vdots & & \vdots & & & \\ \bar{a}_{m1} & \bar{a}_{m2} & \ldots & \bar{a}_{mn} & & & 1 \end{pmatrix}$$

Given x then for $i = 1$ to n set the lower bounds as:

$$x_i \geq 0 \Longrightarrow \ell_i = 0$$

$$x_i < 0 \Longrightarrow \ell_i = x_i$$

and for $i = n+1$ to $n+m$ set the lower bounds as:

$$x_i > 0 \Longrightarrow \ell_i = 0$$

$$x_i \leq 0 \Longrightarrow \ell_i = x_i - \rho$$

$$\bar{c}_j = c_j - \pi a^j$$

$$\bar{c}_j = c_j - c_B B^{-1} a^j = c_j - c_B \bar{d}^j$$

Case 1:

$$\text{if} \quad c_j > 0_{j \in N} \Longrightarrow x_j \longleftarrow x_j + \theta$$

$$x_B = x_B - \theta \bar{d}^j$$

if $\bar{d}_i^j < 0 \quad \forall i$ then problem is unbounded,
else we find θ such that:

$$\theta = \theta_{i^*} = \min \left\{ \frac{x_{B_i} - \ell_i}{\bar{d}_i^j}, \quad \bar{d}_i^j > 0 \right\}$$

hence x_j enters the basis and x_{i^*} leaves the basis.

Case 2:

$$\text{if} \quad c_j < 0_{j \in N} \Longrightarrow x_j \longleftarrow x_j - \theta$$

$$x_B = x_B + \theta \bar{d}^j$$

Where $\theta = \min \theta_1, \theta_2$ and $\theta_1 = x_j - \ell_j$ and

$$\theta_2 = \theta_{i^*} = \min \left\{ \frac{\ell_i - x_{B_i}}{\bar{d}_i^j}, \quad \bar{d}_i^j < 0 \right\}$$

If $\theta = \theta_1$ then nonbasic j reachs its lower bound and basic doesn't change. If $\theta = \theta_2$ then x_{i^*} leaves the basis and x_j enters the basis.

vii) Go to step (iv).

We randomly generated some **LP** problems and solved those by our algorithm but the results show that the method is not efficient for solving **LP** problems by using the exterior penalty function $\mathbf{F_2(x)}$. We have also tried primal and dual problems together, but the results do not seem promising.

problem #	solution by lindo	solution via NSO
1	99.179	99.184
2	unbounded	unbounded
3	395.34	369.70
4	72.135	68.30
5	85.71	80.30
6	260.00	260.41
7	123.179	118.73
8	109.434	109.45
9	132.699	132.71
10	70.339	69.667
11	70.87	69.174
12	83.849	83.853
13	198.860	171.08

Table 4.1: $Ax = b\,, \quad A \in \mathcal{R}^{16\times 16}$

problem #	solution by lindo	solution via NSO
1	246.420	245.00
2	88.958	89.17
3	231.291	231.558
4	50.216	50.66

Table 4.2: $Ax = b\,, \quad A \in \mathcal{R}^{26\times 26}$

5. CONCLUSIONS

For the feasibility problem, our non-smooth approach seems to be more efficient, at least in the size range of computations discussed in chapter 4 . In the surrogate constraint methods which are efficient versions of relaxation methods, there is no measure of usefulness of the selected surrogate constraint. As a future work, we consider using our algorithm together with the surrogate constraint methods. In our algorithm the penalty function is reduced exponentially in the first loop of iterations (the first n+m iterations). So implementing our algorithm in the beginning of the surrogate constraint method will probably yield more efficient results.

In the case of implementing our algorithm for LP problems we see that putting the objective function as a penalty term in the so defined non-smooth penalty function has a detrimental effect on the convergence of the algorithm. We worked on some variants of penalty functions to decrease that effect of the objective function by assigning a small penalty weight, but it seems that for LP case the penalty weights must be arranged in a more suitable way to gain an effective algorithm.

BIBLIOGRAPHY

REFERENCES

[1] Agmon S., " The relaxation method for linear inequalities " , *Canadian Journal of Mathematics* 6 (1954), 382-392.

[2] Ben-Tal A. and Zowe J., " Necessary and sufficient optimality conditions for a class of nonsmooth minimization problems " , *Mathematical Programming* 24 (1982), 70-91.

[3] Burke V., " Second order necessary and sufficient conitions for convex composite NDO " , *Mathematical Programming* 38 (1987) 287-302.

[4] Coleman T.F. and Conn A.R., " Nonlinear programming via an exact penalty function: Global Analysis " , *Dept. of Computer Science Tech. report* CS-80-31, July, 1980, University of Waterloo.

[5] Coleman T.F. and Conn A.R., " Second order conditions for an exact penalty function " , *Mathematical Programming* 19 (1980) 178-185.

[6] Conn A.R., " Constrained optimization using a nondifferentiable penalty function " ,*SIAM J. Numer. Anal.* vol. 10, No. 4, September 1973.

[7] Conn A.R., " Linear programming via a nondifferentiable penalty function " , *SIAM J. Numer. Anal.* vol. 13, No. 1, March 1976.

[8] Conn A.R. and Pietrzykowski T., " A penalty function method converging directly to a constrained optimum " , *SIAM J. Numer. Anal.* vol. 14, No. 2, April 1977.

[9] Fletcher R., " Practical methods of optimization " , *John Wiley* , Great Britain 1987.

[10] Goffin J.L., " The relaxion method for solving systems of linear inequalities " , *Mathematics of Operational Research* , vol. 5, No. 3, August 1980.

[11] Hestenes M.R., " Conjugate direction methods in optimization " , *Spinger-Verlag*, New York 1985.

[12] Himmelblau D.M., " Applied nonlinear programming " , *McGraw-Hill*, New York 1972.

[13] Lemarechal C. and Mifflin R.(Eds), " Nonsmooth optimizitation " , *IIASA Proceedings 3*, Pergamon, Oxford 1977.

[14] Motzkin T.S. and Schoenberg I.T., " The relaxation method for linear inequalities " , *Canadian Journal of Mathematics* 6 (1954), 393-404.

[15] O'Neill P.F. and Conn A.R., " Nondifferentiable optimization and worse " , *Dep. of Combinatorics and Optimization* report CS-83-05, March 1983.

[16] Polak E. and Mayne D.Q., " Algorithm models for nondifferentiable optimization " , *SIAM J. Control and Optimization* vol. 23, No. 3, May 1985.

[17] Shor N.Z., " Minimization methods for nondifferentiable optimizationy " , *Springer-Verlag* Berlin, Heidelberg 1985.

[18] Shrijver A. and Grötschel M. and Lovász L., " Geometric algorithms and combinatorial optimization " , *Springer-Verlag* Berlin, Heidelberg 1988.

[19] Telgen J., " On relaxation methods for systems of linear inequalities " , *European Journal of Operational Research* 9 (1982) 184-189.

[20] Wolfe P., " A method of conjugate subgradients for minimizing nondifferentiable functions " , *Mathematical Programming Study* (1975) 145-173.

[21] Womersley R.S., " Local properties of algorithms for minimizing nonsmooth composite functions " , *Mathematical Programming* 32 (1985) 69-89.

[22] Wright S.J., " Local properties of inexact methods for minimizing nonsmooth composite functions " , *Mathematical Programming* 37 (1987) 232-252.

[23] Yang K. and Murty K.G., " Surrogate constraint methods for linear inequalities " , *Department of Industrial and Operations Eng.* University of Michigan, June, 1990.