An Algorithm for Solving Indefinite Quadratic Programming Problems

Siddieg AMAEI
Department of Mathematics, Prince Sattam Bin Abdulaziz University, Alkhair Public Library, Sa’ad Ibn Mu‘adth, Al Kharj Saudi Arabia

Abstract

In this paper, we give in section (1) compact description of the algorithm for solving general quadratic programming problems (that is, obtaining a local minimum of a quadratic function subject to inequality constraints) is presented. In section (2), we give practical application of the algorithm, we also discuss the computational work and performing by the algorithm and try to achieve efficiency and stability as possible as we can. In section (3), we show how to update the QR-factors of $A^O$, when the tableau is complementary, we give updating to the LDLT-Factors of $G^K$. In section (4) we are not going to describe a fully detailed method of obtaining an initial feasible point, since linear programming literature is full of such techniques.

Introduction

In this section we give the detailed outlines of the algorithm of indefinite quadratic programming problems. It references to the numbers of some equations and conditions appeared in the following equations [1-8]:

\[
\min_{x} \lambda^K(x) \quad (1)
\]

and

\[
\begin{bmatrix}
G - A^K & 0 \\
A^K & 0
\end{bmatrix}
\begin{bmatrix}
d^K \\
k^K
\end{bmatrix} =
\begin{bmatrix}
0 \\
\xi
\end{bmatrix}.
\]

(2)

the algorithm assumes the availability of an initial basic feasible point [9-15]. The steps are:

1. Given $\bar{x}^{(t)}$, $\bar{A}^{(t)}$, $v^{(t)}$, and $\eta$, set $K=1$.
2. Solve (1) for $q$.
3. If $\lambda^K_\bar{e} \geq 0$ terminate with $\bar{x}^* = \bar{x}^{(K)}$ otherwise solve $\min_{x \neq \eta} \frac{\lambda^K_\bar{e}}{d^K}$ for $P_r$.
4. If $d\lambda^K_\bar{e} < 0$ and $\frac{\lambda^K_\bar{e}}{d^K} \leq \frac{\lambda^K_\bar{e}}{d^K}$ is satisfied remove $q$ from $\eta$, update the basic variables using [16-20] to $v^K = v^K - d^K$, $P \in \eta \setminus \{q\}$; set $K=K+1$ and go to (b) otherwise remove $q$ from $\eta$ [21-28]; update the basic variables using $v^K = v^K + d^K$, $P \in \eta \setminus \{q\}$.
5. Set $r=1$ and $k=k+1$ and add $P_r$, to $\eta$.
6. Solve $\min_{x \neq \eta} \frac{\lambda^K_\bar{e}}{d^K}$ for $P_r$.
7. If $\sigma d\lambda^K_\bar{e} < 0$ and $\frac{\lambda^K_\bar{e}}{d^K} \geq \frac{\lambda^K_\bar{e}}{d^K}$ is satisfied, update the basic variables using $\lambda^K_\bar{e} = \frac{\lambda^K_\bar{e}}{d^K}$ to $v^K = v^K + d^K$, $P \in \eta$.

Set $K=K+1$ and go to (b).

Other wise update the basic variables using $\lambda^K_\bar{e} = \frac{\lambda^K_\bar{e}}{d^K}$ to $v^K = v^K + d^K$, $P \in \eta$; set $r=r+1$ and go to (e).

Practical Application of the Algorithm

The algorithm presented above represents a general outline of a method for solving indefinite quadratic programming problems rather than an exact definition of a computer implementation. In this section we discuss the computational work performed by the algorithm, and try to achieve efficiency and stability as possible as we can. In doing so we follow, with slight modifications, the work of Gill and Murray which has been applied to active set methods since mid-seventies until now [7-10]. The slight modifications are made to cope with the new forms of the matrices used in the method when G is indefinite. In the case when G is positive (semi definite) the active set methods are considered to be equivalent, [20], pointed out. There he gave a detailed description of that equivalence. He also re-mentioned this equivalence [6]. The major computational work of the algorithm is in the solution of

\[
\begin{bmatrix}
G - A^K & -W & 0 & 0 & 0 \\
-A^K & 0 & 0 & 0 & 0 \\
-A^K & 0 & 0 & 0 & 0 \\
-A^K & 0 & 0 & 0 & 0 \\
-A^K & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
d^K \\
k^K
\end{bmatrix} =
\begin{bmatrix}
0 \\
\xi
\end{bmatrix},
\]

(3)

and

\[
H = Z^K \left( Z^K G^K Z^K \right)^{-1} Z^K T
\]

(4)

\[
T = S^K - Z^K \left( Z^K G^K Z^K \right)^{-1} Z^K G^K S^K
\]

(5)

*Corresponding author: Siddieg AMAEI, Department of Mathematics, Prince Sattam Bin Abdulaziz University, Alkhair Public Library, Sa’ad Ibn Mu‘adth, Al Kharj Saudi Arabia, Tel: +966 11 588 8888, E-mail: wigdan@hotmail.com

Received December 12, 2017; Accepted January 11, 2018; Published January 24, 2018

Citation: Siddieg AMAEI (2018) An Algorithm for Solving Indefinite Quadratic Programming Problems. J Appl Comput Math 7: 380. doi: 10.4172/2168-9679.1000380

Copyright: © 2018 Siddieg AMAEI. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
\[ U = S^{Kf} GZ^{Kf} \left( Z^{Kf} GZ^{Kf} \right)^{-1} Z^{Kf} GS^{Kf} - S^{Kf} GS^{Kf} \]

We do not solve them directly; instead, we make use of the special structure of the matrices involved. We use the matrices \( H, T \) and \( U \) defined in eqn. (5). Thus, accordingly the solution in eqn. (3) is given by:

\[ d_s^{(k)} = -T e_s \]
\[ d_t^{(k)} = U e_t \]
\[ d_u^{(k)} = A_t^{Kf} d_t^{(k)} \]
\[ d_s^{(k+p)} = HW d_s^{(k+p)} + T e_s d_t^{(k+p)} - H \beta \]
\[ d_u^{(k+r)} = -T W d_s^{(k+r)} - U e_t d_t^{(k+r)} + T^T d_t^{(k+r)} \]

Thus, in (21) is updated from the complementary tableau to another [12].

Referring to \((Z^{Kf} GZ^{Kf})^2 = Z^T (G + GZ)\), \( H, T \) and \( U \) are given by:

\[ H = Z^{Kf} - Z^{Kf} \left( Z^{Kf} GZ^{Kf} \right)^{-1} Z^{Kf} \]
\[ T = S^{Kf} - Z^{Kf} \left( Z^{Kf} GZ^{Kf} \right)^{-1} Z^{Kf} GS^{Kf} \]
\[ U = S^{Kf} GZ^{Kf} \left( Z^{Kf} GZ^{Kf} \right)^{-1} Z^{Kf} GS^{Kf} - S^{Kf} GS^{Kf} \]

Where \( S^{Kf} \) and \( Z^{Kf} \) satisfy

\[ S^{Kf} A^{Kf} = I \]
\[ Z^{Kf} A^{Kf} = 0 \]

The choice of \( S^{Kf} \) and \( Z^{Kf} \) to satisfy eqns. (13) and (14), respectively is generally open. Here we take the choice given in S=QR, Z=Q, which is, according to K(Z^{Kf} GZ^{Kf})K(G), is advantageous as far as stability is concerned. For the sake of making this section self-contained we show how \( S^{Kf} \) and \( Z^{Kf} \) are obtained in away suitable to this section.

\[ \hat{Q}^{Kf} A^{Kf} = \begin{bmatrix} R^{Kf} \\ 0 \end{bmatrix} \]

represent the QR factorization of \( A^{Kf} \), where \( A^{Kf} \) is n×n. Thus \( Q^{Kf} \) is n×n and \( R^{Kf} \) is an \( L_k \times L_k \) upper triangular partition of \( Q^{Kf} \). Hence, in eqn. (15) we have:

\[ \hat{Q}^{Kf} A^{Kf} = R^{Kf} \]
\[ Q^{Kf} A^{Kf} = 0 \]

so in eqns. (16) and (17) we define \( S^{Kf} \) and \( Z^{Kf} \) by:

\[ S^{Kf} = \hat{Q}^{Kf} R^{Kf} \]
\[ Z^{Kf} = \hat{Q}^{Kf} T^{Kf} \]

Where the identity matrix is whose columns are reversed. Thus we conclude by saying that the computation is focused on using the QR factorization of \( A^{Kf} \), (when the kth iteration is complementary).

So updating these factors is required at each iteration when the tableau is complementary [25].

**Updating the QR-Factors of** \( A^{Kf} \)**

In this section we show how to update the QR-factors of \( A^{Kf} \), when the tableau is complementary, also we give updating to the LDI-Factors of \( G^{Kf} \).

Following the stream of our discussions, two cases are to be considered separately. The case when the (k+1)th iteration results in a complementary tableau, and the case when complementarity is restored at the (k+r+1)th iteration after r successive non-complementary tableaux. In the first case the factors of \( A^{Kf} \) are updated to give those of \( A^{Kf+1} \), and this is the case when a column, \( g_r \), say, is deleted from \( A^{Kf+1} \). In the second case the factors of \( A^{Kf} \) are used to give those of \( A^{Kf+r+1} \), and this is the case when one column, \( g_r \), is deleted from \( A^{Kf} \) and then r other columns are added to \( A^{Kf} \). We follow the same steps carried [9], with the appropriate modification in the second case. In the first case, let \( A^{Kf} \) be the nx(L-k-1) matrix obtained by deleting the qth column, \( g_r \), from \( A^{Kf} \). Suppose the QR-factorization of \( A^{Kf+r} \) is given by:

\[ Q^{Kf} A^{Kf} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \]

\[ Q^{Kf+1} = \begin{bmatrix} A_1^{Kf} & g_r \\ A_2^{Kf} & A_3^{Kf} \end{bmatrix} \]

Where \( A_1^{Kf} \) is nx(q-1) and \( A_3^{Kf} \) is nx(L-k-q). Let \( R^{Kf} \) have the form

\[ R^{Kf} = \begin{bmatrix} R_{11} & g_r \\ 0 & 0 \end{bmatrix} \]

where \( R_{11} \) is (q-1)×(q-1) upper triangular, \( R_{12} \) is (q-1)×(L-k-q), \( R_{22} \) is (L-k-q)×(L-k-q) upper triangular, \( g_r \) is a (q-L) vector, \( \beta \) is an (L-k-q) vector and \( \gamma \) is a scalar. Since \( A^{Kf+1} = \begin{bmatrix} A_1^{Kf} & A_2^{Kf} \\ A_3^{Kf} & A_4^{Kf} \end{bmatrix} \), \( Q^{Kf+1} A^{Kf+1} \) will have the form:

\[ \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} = R^{Kf} \]

Now, let \( Q^{Kf} \) be the product of the plane rotations which gives:

\[ \begin{bmatrix} \beta^T \\ R_{22} \end{bmatrix} = R^{Kf} \]

where \( R' \) is (L-k-q)×(L-k-q) upper triangular. In this case \( Q^{Kf} \) is an \( (L_k+q+1) \times (L_k+q+1) \) orthogonal matrix. Thus if \( Q^{Kf} = \begin{bmatrix} I_{L_k} & 0 \\ 0 & L_k - q + 1 \end{bmatrix} \),

\[ \begin{bmatrix} I_{L_k} & 0 \\ 0 & L_k - q + 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \]

We obtain \( Q^{Kf} A^{Kf+1} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \)

Thus, only the rows from the qth to the \( L_k \)th of \( Q^{Kf} \) are altered in obtaining \( Q^{Kf+1} \), so if \( K^{Kf} \) is partitioned into:

\[ Q^{Kf} = \begin{bmatrix} Q_{11}^{Kf} & L_{K-k} \end{bmatrix} \]

then \( Q_{11}^{Kf} \) in particular, takes the form:
\[
Q_{k+1}^{(i)} = \begin{bmatrix}
q_i^n \\
q_i^{(i)}
\end{bmatrix}
\]

Note also that the first q-1 rows of \(Q_{k}^{(i)}\) are not changed. This fact might be helpful as far as efficiency is concerned if we want to think of another alternative of choosing \(q\) in eqn. (1), such an alternative is:

\[
Q = \text{max}\{i|\lambda_i < 0, 1 \leq i \leq L_k\}
\]

(23)

So that increase the number of rows of \(Q_{k}^{(i)}\) and \(R_{k}^{(i)}\) which are unaltered in iteration \((K+1)\), which in turns reduces the effort, especially when \(L_k\) is relatively large. We now consider the second case when complementarity is restored at the \((k+r+1)\)th iteration. Let \(W = [\sigma_{n+1}, \sigma_{n+2}, \ldots, \sigma_{n+1}^n]\). Let \(A_{k+1}^{(i)}\) be obtained from \(A_{k}^{(i)}\) by removing \(\sigma_{n+1}\). Thus

\[
A_{k+1}^{(i+1)} = [A_{k+1}^{(i)}; W]
\]

(24)

Pre-multiplying both sides in eqn. (24) by \(Q_{k+1}^{(i)}\) (defined in eqn. (21)) we get

\[
Q_{k+1}^{(i)}A_{k+1}^{(i+1)} = \begin{bmatrix}
W_k^{(i)} \\
0
\end{bmatrix}
\]

(25)

where \(W_k^{(i)} = Q_{k+1}^{(i)}W, W_k' = Q_{k+1}^{(i)}W\) and \(R_{k+1}^{(i)}\) is defined in eqn. (20).

(20)

Let \(QW_k' = \begin{bmatrix}
R_2' \\
0
\end{bmatrix}
\]

(26)

Define the QR-factorization of \(W_k'\). Here \(Q_{k+1}^{(i)}\) is \((h-L_k+1) \times (n-L_k+1)\) and orthogonal, and \(R_2'\) is \(r \times r\) upper triangular. If

\[
Q_{k+1}^{(i)} = \begin{bmatrix}
1 & 0 \\
0 & Q_{k+1}^{(i)}
\end{bmatrix}
\]

Then

\[
Q_{k+1}^{(i)} = \begin{bmatrix}
R_k^{(i)} \\
0
\end{bmatrix}
\]

(27)

Thus we obtain the QR-factorization of \(A_{k+1}^{(i+1)}\) with

\[
Q_{k+1}^{(i+1)} = Q_{k+1}^{(i)}A_{k+1}^{(i+1)} = \begin{bmatrix}
W_k^{(i)} \\
0
\end{bmatrix}
\]

And

\[
R_{k+1}^{(i+1)} = \begin{bmatrix}
R_2' \\
0
\end{bmatrix}
\]

(28)

Updating The LDL\(^{-1}\)-Factors of \(G_{k}^{(i)}\)

The factors \(L^{(k)}D^{(k)}L^{(k)^T}\) of \(G_{k}^{(i)}\) are updated at each iteration when the tableau is complementary. Near the end of this section we show that \(G_{i}^{(i)}\) is always positive definite (on the assumption that \(G_{k+1}^{(i)}\) is positive semi-definite). Updating these factors is very stable when \(L_k\) is relatively large. We now consider the case when the \((k+1)\)th iteration results in a complementary tableau. Unfortunately, in the other case when complementarity is restored at the \((k+r+1)\)th iteration, we are unable till now to explore a way of using the factors of \(G_{k}^{(i)}\) in obtaining those of \(G_{k+1}^{(i+1)}\). However \(n-L_k\), the dimension of \(G_{k}^{(i+1)}\), decreases with \(r\), in which case the effort of re-factorizing \(G_{k}^{(i+1)}\) might not be so much, especially when \(n-L_k\) is itself small.

This calls for choosing the starting \(L_k\) so that \(n-L_k\) is small. In the case when the number of constraints is greater than \(n-L_k\), \(L_k\) is chosen to be equal to \(n\); that is the initial guess \(L_k^{(i)}\) is a vertex. With this choice \(G_{i}^{(i)} = 0\), and in the second iteration we might expect a constraint to be deleted from the active set (which is the case when the second iteration is complementary). Otherwise the third iteration will definitely restore complementarity at another vertex leaving \(G_{k+1}^{(i)} = 0\). In the former case the dimension of \(G_{k+1}^{(i)}\) is \(1\). In general the dimension of \(G_{k+1}^{(i)}\) keeps on increasing when constraints are deleted, and updating the factors is straight forward as will be shown. On the other hand the dimension of \(G_{k}^{(i)}\) keeps on decreasing when constraints are added to the active set, and in this case we are faced with re-factorizing the factors. We return to the case when the \((k+1)\)th iteration is complementary. Here we are almost copying the work of in eqn. (9). In this case, as in eqn. (25) shows \(G_{k+1}^{(i)} = \begin{bmatrix}
q_i^n \\
q_i^{(i)}
\end{bmatrix}
\)

(37)

and in the second iteration we might expect a constraint to be

\[
Z^{(k+1)} = I G_{k+1}^{(i)} = \begin{bmatrix}
Z^{(k)}q
\end{bmatrix}
\]

(30)

the matrix \(G_{k+1}^{(i)}\) is given by:

\[
G_{k+1}^{(i)} = \begin{bmatrix}
G_{k}^{(i)}Z^{(i)}G_{k}^{(i)}
\end{bmatrix}
\]

(31)

It can be shown that when a symmetric matrix is augmented by a single row and a column, the lower-triangular factor is augmented by a single row. Define:

\[
L_{k}^{(i)} = \begin{bmatrix}
L_{k}^{(i)} & 0 \\
0 & 1
\end{bmatrix}
\]

\[
D_{k}^{(i)} = \begin{bmatrix}
D_{k}^{(i)} & 0 \\
0 & d_{k+1,i+1}
\end{bmatrix}
\]

(32)

If we substitute in eqn. (31) and in eqn.(32) into the identity:

\[
G_{k+1}^{(i)} = L_{k+1}^{(i)}D_{k+1}^{(i)}L_{k+1}^{(i)^T},
\]

we obtain \(L_{k}^{(i)}\) and \(d_{k+1,i+1}\) as the solution of the equations

\[
L_{k}^{(i)}D_{k}^{(i)}L_{k}^{(i)^T} = Z^{(i)}G_{k}^{(i)}
\]

(33)

and

\[
d_{k+1,i+1} = q_i^n G_{k}^{(i)} - L_{k}^{(i)^T}D_{k}^{(i)}L_{k}^{(i)}
\]

(34)

The numerical stability of this scheme is based on the fact that, if \(G_{k+1}^{(i)}\) is positive definite, the element \(d_{k+1,i+1}\) must be positive. In this event in eqn. (34) ensures that arbitrary growth in magnitude cannot occur in the elements of \(L_{k}^{(i)}\).

Before ending this section we show that when the \(k\)th iteration and the \((k+1)\)th iteration are complementary then \(G_{k+1}^{(i)}\) must be positive definite.

Let the tableau be complementary at the \(k\)th iteration. Let \(A_{k}^{(i)}\) be the matrix whose columns correspond to the active constraints, and \(\lambda^{(i)} < 0\). The increase of \(v_{q}\) changes \(f\) according to

\[
f = f^{(k)} + \lambda^{(i)} v_{q} = 0.5u_{q} v_{q}^{T}
\]

(35)

and \(\lambda^{(i)} = \lambda^{(i)} - u_{q} v_{q}^{T}\)

(36)

and \(\lambda^{(i)}\) changes according to

\[
\lambda^{(k+1)} = \lambda^{(k)} + T_{g}^{(k+1)} v_{q}
\]

(37)

For the next tableau (i.e., the \((k+1)\)th) to be complementary \(u_{q}\) must be negative, and the new value \(v_{q}^{(k+1)} = \frac{\lambda^{(k)}}{u_{q}^{(k)}}\) of \(v_{q}\) must not violate feasibility.

Thus, using in eqn. (35), we have \(d_{q}^{(k+1)} = -u_{q} > 0\) which reflects the fact that \(f\) possesses a positive curvature along the direction \(T_{g}^{(k+1)}\).
Now let $A^{(k+1)}$ be obtained from $A^{(k)}$ by removing $a_i$, and let $Z^{(k+1)}$ be defined so that $Z^{(k+1)^T}d^{(k+1)} = 0$.

Pre-multiply both sides of in eqn. (37) by $A^{(k+1)^T}$ to get:

$$A^{(k+1)^T}T_{\mathbf{v}} = 0$$

(38)

showing that $T_{\mathbf{v}}$ lies in the space spanned by the columns of $Z^{(k+1)}$, so

$$T_{\mathbf{v}} = Z^{(k+1)^T}h$$

(39)

for some $(n-L_{k+1})$ vector $h$. Since along $T_{\mathbf{v}}$, at $v^{(k+1)}_q$, $df\over dv_q = 0$ and $d^2f\over dv_q^2 > 0$, then $f$ is minimum along $T_{\mathbf{v}}$ at $v^{(k+1)}_q$, where

$$x^{(k+1)}_j = x^{(k)}_j + T_{\mathbf{v}}v^{(k+1)}_j$$

(40)

We therefore conclude, in the active set methods sense, that the direction $d^{(k+1)}_A = T_{\mathbf{v}}v^{(k+1)}_q$ solves the equality problem:

minimize $0.5\delta^2_q \bar{G}_A\delta_q + \delta^2_q \bar{G}_A(G_k^{(k)} + g)$

Subject to $A^{(k+1)}\delta_q = 0$

(41)

Using (39), $d^{(k+1)}_A = Z^{(k+1)^T}h v^{(k+1)}_q = Z^{(k+1)^T}d^{(k+1)}_A$, thus $d^{(k+1)}_A$ solves the problem

minimize $0.5\delta^2_q (Z^{(k+1)^T}G(Z^{(k+1)}_A)\delta_q + d^2_q (G_k^{(k)} + g))$

from which we conclude that $G^{(k+1)}_A = Z^{(k+1)^T}G(Z^{(k+1)}_A)$ is positive definite.

Finding an Initial Feasible Point

In this section we are not going to describe a fully detailed method of obtaining an initial feasible point, since linear programming literature is full of such techniques. The method of finding a feasible point has been resolved in linear programming by a technique known as phase 1 simplex [27]. The basis of the technique is to define an artificial objective function, namely: $\bar{F}(x) = -\sum_{j \in \mathbf{S}} \bar{x}_j b_j$, where $v(x)$ is the set of indices of constraints which are violated at the point $x$, and to minimize this function with respect to $x$, subject to the constraints $\bar{x}_j b_j \geq 0$, $j \notin v(x)$. The function $\bar{F}(x)$ is linear and is known as the sum of infeasibilities. If a feasible point exists it is such that $\bar{F}(x) = 0$. In the case when $m$ exceeds $n$, a non-feasible vertex is available as an initial feasible point to phase 1 and the simplex method is applied to minimize $\bar{F}(x)$. This process will ultimately lead to a feasible vertex [28]. Direct application of this method to finding a feasible point in the case when $m$ is less than $n$ is not feasible since, although a feasible point may exist a feasible vertex will not. Under these circumstances artificial vertices may be defined by adding simple bounds to the variables, which could lead to either a poor initial point, since some of these additional constraints must be active, or exclusion of the feasible region. A way out of this dilemma is described [6-9] a number of methods including the above one have been described. Gill and Murray is advantageous in that it makes available the QR-factorization of the initial matrix of active constraints which is then directly used in our algorithm.

References

1. Bazaraa SM, Sherali DH, Shetty CM (1994) Nonlinear Programming: Theory and Algorithm.
2. Coleman LLS (1990) Numerical Optimization. SIAM Books.
3. Cottle RW (1990) The Principle Pivoting method positive visited math program. 48: 369-385.
4. David GL (2003) Linear and nonlinear programming (2nd edn.), Pearson Education.
5. Dennis, Schnabel (1998) Numerical Methods for unconstrained Optimization and nonlinear equation classics in applied Mathematics. Society for Industrial and Applied Mathematics.
6. Fletcher R (2013) Practical Methods of Optimization. (2nd edn.), John Wiley and Sons.
7. Gill PE, Murray W (1973) A numerically stable form of the simplex Algorithm. Journal Linear Algebra Appliers 7: 99-138.
8. Gill PE, Murray W (1975) Numerical Methods for constrained optimization. Academic press.
9. Gill PE, Murray W (1978) Numerically Stable methods for quadratic programming. Math Programme 14: 349-372.
10. Gill PE, Murray W, Margaret W (1981) Practical Optimization. Academic Press.
11. Kunisch K, Rendl F (2003) An infeasible active set method for quadratic problems with simple bounds. SIAM Journal on Optimization 14: 35-52.
12. Mohsin HAH (1996) An Extension to the Dantzig-Wolfe Method for general quadratic programming. University of Khartoum.
13. Byrd RH, Gilbert JC, Nocedal J (2000) A trust region method for nonlinear programming. 9: 149-185.
14. Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University.
15. Jorge N (2006) Stephen Wright books. Springer Series in Operations Research and Financial Engineering.
16. Stephen G N, Ariela S (1996) Linear and non -liner programming. McGraw Hill, New York.
17. Anitescu M (2005) On solving mathematical Programs with complementarity constraints as nonlinear programs. SIAM Journal on Optimization 15: 1203-1236.
18. Byrd RH, Hribar ME, Nocedal J (1999) An interior point algorithm for large-scale nonlinear programming. Society for Industrial and Applied Mathematics 14: 777-900.
19. Gould NIM, Toint PL (2005) An interactive working set method for large scale nonlinear optimization, Acta Numerica 14: 299-361.
20. Gould NIM, Toint PL (2002) An iterative working set method for large scale non-convex quadratic programming. Applied Numerical Mathematics 43: 109-128.
21. Fletcher R, Leyffer S (2002) Nonlinear programming without a penalty function. Mathematical Programming 91: 239-269.
22. Gill PEM, Murray W, Wright M (1991) Numerical Linear Algebra and Optimization.
23. Kočvara M, Stingl MP (2003) PENNON: A code for non-convex non-linear and semi-definite programming Optimization Methods and software. 18: 317-333.
24. Vanderbei RJ, Shanno DF (1999) An interior point algorithm for non-convex non-linear programming. Computation and Applications 13: 231-252.
25. Vavasis SA (1990) ‘Quadratic programming is NP. Information Processing Letters 36: 73-77.
26. Gill PEM, Murray W, Wright MH (1981) Practical Optimization. Academic Press.
27. Vavasis SA (1991) Nonlinear Optimization. Oxford University Press, New York and Oxford.
28. Higham NJ (1996) Accuracy and Stability Of Numerical Algorithms. SIAM Publications Philadelphia.