Comm. Korean Math. Soc. 9 (1994), No. 3, pp. 753–765

ITERATIVE METHODS FOR LARGE-SCALE CONVEX QUADRATIC AND CONCAVE PROGRAMS

SEYOUNG OH

1. Introduction

The linearly constrained quadratic programming(QP) considered is:

$$\begin{aligned} \min \quad & f(x) = c^T x + \frac{1}{2} x^T H x \\ \text{subject to} \quad & A^T x \geq b, \end{aligned} \tag{1}$$

where $c, x \in R^n$, $b \in R^m$, $H \in R^{n \times n}$, symmetric, and $A \in R^{n \times m}$. If there are bounds on x, these are included in the matrix A^T. The Hessian matrix H may be positive definite or negative semi-definite. For large problems H and the constraint matrix A are assumed to be sparse.

The quadratic programming problems, a generalization of linear programming, may arise naturally in economics, planning, and engineering design. In addition, the general nonlinear problem can be solved by posing a sequence of quadratic problems approximated, or by simplifying it into a quadratic problem, see [HIM72].

This problem has been investigated extensively, and good computational methods proposed for its solution in [GIL90], [GIL91], [GOU89], and [PON91]. Many interior-type algorithms have been developed for the convex and the nonconvex cases: an extended Karmarkar's projective algorithm([KAP86], [YEE89]), path-following algorithm([DAY88], [GOL91], [KOJ89], [MEH90], [MON89]), and affine scaling algorithm ([YE92]).

In this paper an efficient computational algorithm for each of these two cases(convex QP and general concave problem) will be described. The methods are both closely related, and are based on an active set strategy

Received February 16, 1994.

and solving the corresponding KKT equations at each iteration. For each type of problem a Karush-Kuhn-Tucker (KKT) point will always be found. In the positive definite case this will, of course, be a unique global minimum. In the other case there will, in general, be many KKT points. The algorithm will always find one of these, which will almost always be a constrained local minimum with a lower function value than the initial feasible point. For some initial points the global minimum will be found.

In Section 2, a complete description of the algorithm for the case where H is positive definite is given.

The concave quadratic problem is a special case of minimizing a differentiable concave function $f(x)$ subject to linear inequality constraints. The algorithm will be given in terms of this more general problem in Section 3.

Our purpose is to develop algorithms which will be efficient for large sparse quadratic problems, and that has motivated the approach presented here. As now implemented however, they do not take advantage of a sparse matrix structure. Modification for sparse matrix problems should not be difficult. The algorithms, described in what follows, depend essentially on the solution of a single large sparse system of linear equations (the KKT system) at each iteration. This linear system usually changes only in one row and column between successive iterations, and the method of solution takes advantage of this fact.

Test problems as large as $n = 160$, $m = 641$ were generated and solved using the CRAY Y-MP. Results are presented in Table 1-4 and discussed in Section 4. For the comparison purpose all problems were also run using MINOS 5.3 [MUR83]. The average time for the KKT algorithm was 6.2 to 12.8 times faster than MINOS over the range of problems tested. Also the average number of iterations required was usually 2–4 times more for MINOS.

2. KKT Algorithm for H Positive Definite

In this section we give a detailed description of the algorithm which finds the unique KKT point for the quadratic problem (1) with H positive definite.

In order to deal with individual constraints we let a_i be the i^{th} column

of A, and define the value of the i^{th} constraint as

$$\phi_i = a_i^T x^k - b_i \ , \quad i = 1, \ldots, m \ . \tag{2}$$

The feasible set is then given by

$$S \ = \ \{ \ x \mid \phi_i(x) \ \geq \ 0 \ , \quad i = 1 \ldots, m \} \ . \tag{3}$$

Corresponding to any x is an active set of constraints:

$$J_A(x) \ = \ \{ \ i \mid \phi_i(x) = 0 \ \}, \quad |J_A| \ \leq \ m \ . \tag{4}$$

A *working set*, a set of selected columns of A in the current iteration, is a subset $J_W(x) \subseteq J_A(x)$, with $|J_W| = q \leq n$, such that $\text{rank}(A_W) = q$, where $A_W = [a_i]_{i \in J_W}$.

Corresponding to any working set J_W, is an equality constrained quadratic program

$$\begin{aligned} \min \quad & c^T x + \frac{1}{2} x^T H x \\ \text{subject to} \quad & A_W^T x = b_W, \end{aligned} \tag{5}$$

where $b_W \in R^q$ consists of the elements b_i of b corresponding to $i \in J_W$. The unique solution $(x^W, \ \lambda^W)$ to (5) is given by the KKT system

$$\begin{pmatrix} H & A_W \\ A_W^T & 0 \end{pmatrix} \begin{pmatrix} x \\ -\lambda \end{pmatrix} = \begin{pmatrix} -c \\ b_W \end{pmatrix}.$$

The vector $\lambda^W \in R^q$, consists of the multipliers associated with the equality constraints in (5). For an arbitrary choice of the working set, it will generally be the case that $x^W \notin S$, or $\lambda^W \ngeq 0$, or both. However, there is always at least one choice of working set such that $x^W \in S$ and $\lambda \geq 0$. When such a choice is made, the point x^W is the unique solution of the original inequality constrained problem (1), since it is the KKT point for that problem. Thus we can think of the algorithm as a systematic search for the correct working set.

If we denote by J_k the working set at the k^{th} iteration, $A_k = [a_i]_{i \in J_k}$, and b_k the corresponding vector to b_W, the algorithm starts with $J_0 = \emptyset$,

so that x^0 satisfies $Hx^0 = -c$. If $x^0 \in S$, then x^0 solves (1). For the k^{th} iteration, let $(x^k,\ \lambda^k)$ be the solution of the linear system (6) with $A_W = A_k$ and $b_W = b_k$. If $x^k \in S$ and $\lambda^k \geq 0$, then x^k solves (1). Otherwise, $\phi_i(x^k) < 0$, for at least one value of i, or $\lambda_j^k < 0$ for at least one value of j, or both. The working set is modified by adding to J_k the index i corresponding to the most negative $\phi_i(x^k)$, deleting from J_k the index j corresponding to the most negative λ_j^k, or both. This gives a new working set J_{k+1} and corresponding KKT system (6), for iteration $k+1$.

To simplify the notation, we write (6) at the k^{th} iteration as

$$W_k \begin{pmatrix} x \\ -\lambda \end{pmatrix} = \begin{pmatrix} -c \\ b_k \end{pmatrix}, \quad W_k = \begin{pmatrix} H & A_k \\ A_k^T & 0 \end{pmatrix}, \tag{7}$$

This system is solved by an LU factorization of the matrix W_k and subsequent backsolve. After the initial factorization of H, it is only necessary to update or downdate the LU factorization at each iteration.

A difficulty can occur if $q = n$, $\lambda^k \geq 0$, and $x^k \notin S$, since that gives $|J_{k+1}| = n+1$. The number of variables is then increased to n+1, by adding a slack x_s and a corresponding penalty term $\frac{1}{2}\mu x_s^T x_s$ to force the slack to zero with μ sufficiently large. In fact, this may occur several times in succession, so that more than one slack variable may be required. This is discussed further at the end of this section.

ALGORITHM 2.1. KKT Algorithm for quadratic problem with H positive definite.

1. Solve $Hx^0 = c$ for x^0. Set $k = 0$ and $J_0 = \emptyset$, $\lambda^0 = 0$.
2. Compute $\phi_s(x^k) = \min_{i \notin J_k} \phi_i(x^k)$ and $\lambda_t^k = \min_{i \in J_k} \lambda_i^k$.
3. If $\phi_s(x^k) \geq 0$ and $\lambda_t^k \geq 0$, then $(x^k,\ \lambda^k)$ is optimal; stop.
4. If $\phi_s(x^k) < 0$, add s to working set; $J_k := J_k + \{s\}$. If there is a tie, so that $\phi_i = \phi_s$ for more than one i, we choose i so that $\nabla f(x^k)^T a_i$ is a maximum.
5. If $\lambda_t^k < 0$, drop t from working set; $J_k := J_k - \{t\}$.
6. If $|J_k| > n$, add a slack variable x_s and a penalty term $\frac{1}{2}\mu x_s^T x_s$ to the objective function. $n := n+1$.
7. Update the LU factorization of the KKT matrix W_k with the added or/and the deleted constraints. Then backsolve (7) for $(x^k,\ \lambda^k)$, $k := k+1$, and go to 2.

- Addition of Slacks

As discussed earlier, one or more, slack variables may be needed (thereby increasing n) so that $|J_k| \leq n$. To handle this situation we increase n by adding s slack variables, where $s = q - n$ (usually $s = 1$). Now assume the active constraints in the k^{th} iteration given by $A_q x = b_q$ where A_q is a $q \times n$ matrix and $q > n$. The partition of A_q and b_q as follows:

$$A_q^T = \begin{pmatrix} A_n^T \\ A_s^T \end{pmatrix}, \quad b_q = \begin{pmatrix} b_n, \\ b_s \end{pmatrix}, \tag{8}$$

where $A_n^T \in R^{n \times n}$ has rank n and $A_s^T \in R^{s \times n}$. Now add a slack vector $x_s \in R^s$ to (8), in last s equations, then $A_q^T x = b_q$ becomes

$$\begin{aligned} A_n^T x \; + \; 0 \; x_s &= b_n \\ A_s^T x \; + \; I_s x_s &= b_s \; . \end{aligned}$$

Note that by the rank n of A_n, the $q \times q$ coefficient matrix is nonsingular. To reduce x_s to zero the term $\frac{1}{2}\mu x_s^T x_s$ is added to the objective function, where $\mu \geq 100\|H\|$. Also partition $\lambda = \begin{pmatrix} \lambda_n \\ \lambda_s \end{pmatrix}$ corresponding to the previous partition. This gives the following system of $2q$ equations for the KKT point:

$$\begin{bmatrix} H & 0 & A_n & A_s \\ 0 & \mu I_s & 0 & I_s \\ A_n^T & 0 & 0 & 0 \\ A_s^T & I_s & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ x_s \\ -\lambda_n \\ -\lambda_s \end{bmatrix} = \begin{bmatrix} -c \\ 0 \\ b_n \\ b_s \end{bmatrix} . \tag{9}$$

With the addition of this s slacks, a rank s update of the LU factorization of W_k is needed.

3. KKT Algorithm for H Negative Semidefinite

The quadratic problem (1) for H negative semidefinite is obviously a special case of the following problem:

$$\min_{x \in S} \quad f(x) \; , \tag{10}$$

where $f(x)$ is a differentiable concave function. The key property of this problem is that every local minimum (and therefore the global minimum) is attained at an extreme point of S, see [PAR87]. The algorithm is motivated by this property. The algorithm will be described in terms of this general concave minimization problem.

For the concave minimization algorithm, we generate a descent direction by projecting the steepest descent direction $-\nabla f(x)$ on the subspace spanned by the working set constraints. The step length is limited by a trust region quadratic penalty. This can be done by solving the following strictly convex quadratic problem based on the current feasible point x^k.

$$\begin{aligned} \min \quad & \nabla f(x^k)^T x + \frac{1}{2}\theta(x - x^k)^T(x - x^k) \\ \text{subject to} \quad & A_q^T x = b_q, \end{aligned} \tag{11}$$

where θ is a parameter and A_q is the $n \times q$ coefficient matrix of the working set. Here we choose the value of θ so that the term $\nabla f(x^k)$ dominates θx^k in the objective function of problem (11):

$$\frac{1}{2}\theta\|x\|^2 \; + \; (\nabla f(x^k) - \theta x^k)^T x \; + \; \frac{1}{2}\theta\|x^k\|^2.$$

That is, θ is taken as

$$\theta \; = \; \eta \; \frac{\|\nabla f(x^k)\|}{\|x^k\|} \; ,$$

where η is an appropriate small number (0.1 or 0.01). Since the problem (11) is a strictly convex programming problem, the unique minimum point is obtained by solving a KKT system:

$$\begin{bmatrix} \theta I_n & A_q \\ A_q^T & 0 \end{bmatrix} \begin{bmatrix} x \\ -\lambda \end{bmatrix} \; = \; \begin{bmatrix} \theta x^k \; - \; \nabla f(x^k) \\ b_q \end{bmatrix}. \tag{12}$$

The solution $\overline{x}$ of the above system (12) might not be feasible. So in order to make sure that we are moving in a feasible descent direction, a linear interpolation is used to get a new feasible point x^{k+1}. This new point lies in the intersection of the working set and one more constraint s which the projected direction of $-\nabla f(x^k)$ hits. The linear interpolation

will be shown on later in this section. Now the constraint s is added into the working set J_k. Since it is known that the global minimum function value (and in fact any local minimum value) is attained at a vertex of the feasible polytope, the cardinality of J_k at termination is $|J_k| = n$. In fact, starting with any feasible point x^0, one constraint will be added to the working set at each step, until $|J_k| = n$, so that at most n steps are needed to obtain a vertex. From then on the steps are between adjacent vertices as in the simplex method for linear programming.

Note that the Lagrange multipliers λ_i which are obtained by solving (12) are not for x^{k+1} but for $\bar{x}$, if $x^{k+1} \neq \bar{x}$. Since we know that the solution is at a vertex, a constraint is added at each iteration until $|J_k| = n$. At that time the nonsingular linear system:

$$A_q \lambda = \nabla f(\bar{x}), \quad A_q \in R^{n \times n} .$$

Now the constraint corresponding to the most negative λ can be dropped to decrease the objective function value by making a feasible move off the constraint intersection vertex.

ALGORITHM 3.1. KKT Algorithm for Differentiable Concave Function.

1. Choose any initial feasible point x^0. Set $J_0 = \emptyset$, $k = 0$.
2. If $|J_k| = n$, solve the linear system (13) for λ.
   - If $\lambda \geq 0$, then stop. x is optimal.
   - If $\lambda \not\geq 0$, then drop a constraint corresponding to the most negative λ.
3. Compute $\nabla f(x^k)$, $\theta = \eta \frac{\|\nabla f(x^k)\|}{\|x^k\|}$ with $\eta = 0.01$. Solve the linear system (12) for $\bar{x}$.
4. If $\bar{x}$ is feasible, $x^k = \bar{x}$. Otherwise do a linear interpolation to get a feasible point x_α from (14) and a corresponding constraint s such that

$$\phi_s(x_\alpha) = 0, \qquad \phi_s(\bar{x}) < 0,$$

   and set $x^k = x_\alpha$.
5. Add the constraint s into the working set.
   $k := k + 1$, and go to 2.

In every iteration, θ changes depending on $\|\nabla f(x^k)\|$ and $\|x^k\|$, but the submatrix $\theta I_n \in R^{n\times n}$ of the coefficient matrix of system (12) is just a diagonal matrix. The determination of a initial feasible point will be described later in this section.

- Linear Interpolation

Suppose the solution $\overline{x}$ of KKT system (12) is not feasible. then there must be one or more violated constraint by $\overline{x}$. Let

$$I = \{ i \mid \phi_i(\overline{x}) = a_i^T \overline{x} - b_i < 0 \}$$

and x^k is a feasible point obtained in the k^{th} iteration of this algorithm. Note that $\phi_i(x^k) \geq 0$, $i \in I$. Let $x_\alpha = (1-\alpha)x^k + \alpha\overline{x}$. Then α can be found so that $\phi_s(x_\alpha) = 0$ for some $s \in I$ and x_α is feasible, that is,

$$\alpha = \min_i \frac{\phi_i(x^k)}{\phi_i(x^k) - \phi_i(\overline{x})} < 1, \quad i \in I. \tag{14}$$

Note that $0 < \alpha < 1$ and $\phi_i(x_\alpha) \geq 0$ for all $i = 1, \ldots, m$. Now we can take x_α as a new feasible point x^k which lies in the intersection of the current working set and the constraint s to be added into the working set.

- Determination of a initial feasible point

The KKT algorithm for quadratic problem with H negative definite discussed above requires a feasible initial point x^0 to move to another feasible point by decreasing the objective function value. There are many ways to obtain a initial feasible point, for example, the phase I of simplex method, and see [GIL73], [GIL74]. In addition, considering the algorithm for quadratic problem with H positive definite in Section 2, which does not require any initial feasible point, we can apply it to quadratic problems with H negative definite by using any different strictly convex quadratic objective function subject to the same constraints.

- Updating the KKT system after a change of the working set

When a constraint has been added to or deleted from the working set, it is not necessary to recompute the any factorization of the KKT

system to solve it for a new point. We need $O((n+q)^2)$ flops to update the LU factorization of KKT system when one constraint is added to the working set and at most $O((q-1)^3)$ flops are needed to update the KKT system for a constraint to be deleted from the working set. When one constraint is deleted and another one is added to the working set, the total flops for updating basis matrix is $O((q-1)^3 + (n+q)^2)$.

4. Computational Results

In this section we summarize the computational results obtained when the previously described KKT algorithms were tested on a range of randomly generated test problems. These computational results are presented in Tables 1–4.

Test problems for the two different KKT algorithms (H positive definite and negative definite) had similar polyhedral constraints, except that the constraint set for the positive definite problems were not necessarily bounded (since the solution is always bounded). The coefficients of the dense matrix A consisted of randomly chosen integers from [-99, 99]. The right hand side vector b was then selected so that the point (1.0, 1.0, ..., 1.0) was feasible. In order to insure a bounded feasible set for the negative definite case, the constraints $x_i \geq 0$ and $\sum_i x_i \leq \eta$, were added, where η was chosen sufficiently large so that at least one of the random constraints was active at the minimum point found.

Table 1: Summary of results with KKT algorithm for H positive definite.

Problem Size		No. of Iterations			# Active constraints		CPU Time (sec)		
n	m	Min.	Max.	Avg.	Min.	Max.	Min.	Max.	Avg.
20	80	21	57	31.2	16	20	9.02E-3	4.15E-2	1.74E-2
40	80	31	43	35.7	27	36	1.98E-2	3.85E-2	2.57E-2
40	160	47	72	55.0	34	39	5.72E-2	1.40E-1	8.57E-2
80	160	65	82	72.4	59	69	1.37E-1	2.88E-1	2.05E-1
80	320	98	125	112.3	71	78	4.85E-1	9.05E-1	7.13E-1
160	320	145	173	154.9	125	140	1.31E+0	2.19E+0	1.69E+0
160	640	223	267	238.5	151	157	5.49E+0	7.90E+0	6.14E+0

With no loss of generality we can choose H as a diagonal matrix since if H is not diagonal it can always be diagonalized by a suitable linear transformation. For H positive definite the diagonal elements were randomly chosen in [1, 99]. For H negative definite the elements

were randomly chosen in [-99, 0]. The elements of c in both cases were randomly chosen in [-99, 99].

Table 2: Summary of results with MINOS 5.3 for H positive definite.

Problem Size		No. of Iterations			CPU Time (sec)			Ratio†
n	m	Min.	Max.	Avg.	Min.	Max.	Avg.	
20	80	50	69	58.2	9.51E-2	1.39E-1	1.24E-1	7.1
40	80	95	119	109	2.72E-1	3.62E-1	3.30E-1	12.8
40	160	131	162	146.8	6.40E-1	8.65E-1	7.33E-1	8.6
80	160	228	325	266.2	2.12E+0	3.00E+0	2.40E+0	11.7
80	320	347	445	397.2	4.68E+0	5.96E+0	5.42E+0	7.6
160	320	555	677	615.8	1.35E+1	1.67E+1	1.54E+1	9.1
160	640	981	1098	1040	3.57E+1	3.94E+1	3.81E+1	6.2

† : Ratio = $\frac{T_{MINOS}}{T_{KKT}}$

We now discuss the computational results obtained using the both KKT algorithms on the corresponding set of test problems. In Table 1 the number of iterations, the number of active constraints at optimal, and the time (in CPU seconds) on the CRAY Y-MP, are given as a function of problem size for H positive definite. Each row of the table represents 10 randomly generated problems of the same size. The minimum and maximum values, attained over the 10 problems, of these three quantities are given. The average value for iterations and time is also given.

Table 3: Summary of results with KKT algorithm for H negative definite.

Problem Size		No. of Iterations			# Active constraints		CPU Time (sec)		
n	m	Min.	Max.	Avg.	Min.	Max.	Min.	Max.	Avg.
10	21	10	16	12.9	10	10	1.94E-3	4.28E-3	3.07E-3
10	41	11	20	14.5	10	10	3.30E-3	7.24E-3	4.82E-3
20	41	22	45	29.7	20	20	1.06E-2	3.71E-2	1.94E-2
20	81	26	38	33.5	20	20	1.84E-2	3.64E-2	2.99E-2
40	81	44	88	67.6	40	40	5.99E-2	2.54E-1	1.69E-1
40	161	61	100	82.7	40	40	1.54E-1	3.64E-1	2.76E-1
80	161	151	301	203.7	80	80	1.95E+0	5.62E+0	3.23E+0
80	321	190	312	250.5	80	80	3.06E+0	6.21E+0	4.61E+0
160	321	455	957	767.2	160	160	3.31E+1	8.68E+1	6.40E+1
160	641	634	950	780.0	160	160	5.41E+1	8.23E+1	6.75E+1

The most significant observation is that the number of iterations needed to solve the problem is always less than the number of constraints m. In fact the number of iterations is always less than $2n \leq m$. Recall that this KKT algorithm, for H positive definite, does not need to find a feasible starting point as some other methods do (such as MINOS). In the worst case the number of iterations could grow exponentially with n (as in linear programming). If this approximate linear dependence holds for large sparse problems this algorithm should be very efficient, since the main effort is then equivalent to solving a single large sparse linear system. In Table 2 the results obtained using MINOS 5.3, with the identical set of test problems, are shown. The same solution was obtained by both algorithms for each of the 70 problems. It is seen that both the number of iterations and CPU time are significantly greater for all problems. The average time for MINOS is from 6.6 to 12.8 times greater than for the KKT algorithm.

The results for H negative definite are presented in Tables 3 and 4. The format of these tables is identical to the corresponding tables for H positive definite. Comparing the results we see that, as expected, these concave minimization problems are considerably more difficult than the convex quadratic problems. They share with linear programming the property that every local minimum is attained at an extreme point of the polytope, so that the number of active constraints at x^* is always n.

Table 4: Summary of results with MINOS 5.3 for H negative definite.

Problem Size		No. of Iterations			CPU Time (sec)			Ratio†
n	m	Min.	Max.	Avg.	Min.	Max.	Avg.	
10	21	4	20	10.3	6.01E-3	1.56E-2	9.56E-3	3.1
10	41	13	23	17.9	1.43E-2	2.88E-2	2.09E-2	4.3
20	41	15	28	22.2	2.05E-2	4.01E-2	3.11E-2	1.6
20	81	35	61	43.3	7.33E-2	1.78E-1	1.06E-1	3.5
40	81	47	110	69.8	1.46E-1	4.52E-1	2.53E-1	1.5
40	161	109	188	135.3	7.43E-1	1.13E+0	8.64E-1	3.1
80	161	232	379	298.0	1.53E+0	2.70E+0	2.13E+0	0.7
80	321	397	553	492.8	4.61E+0	6.38E+0	5.61E+0	1.2
160	321	1329	1958	1656	1.74E+1	2.83E+1	2.25E+1	0.4
160	641	1490	2158	1771	4.25E+1	5.95E+1	4.90E+1	0.7

† : Ratio = $\frac{T_{MINOS}}{T_{KKT}}$

It is seen from Table 3 that the number of iterations needed by the KKT algorithm is 2 - 4 times greater than those needed for the convex problems. The number of iterations needed by MINOS (Table 4) was greater than the number needed by the KKT algorithm, but was a relatively smaller increase relative to the convex problems. For these concave problems the actual time required by MINOS was, in several cases, less than that required by the KKT algorithm, as seen by the last column of Table 4. This is because the time per iteration has increased significantly for the KKT algorithm relative to the convex case. This is due primarily to the fact that several complete solutions of the KKT equations are needed rather than a single solution with updating, as in the convex case. A way of avoiding this requirement is being investigated.

References

1. M. B. Daya and C. M. Shetty, *Polynomial barrier function algorithms for convex quadratic programming*, Report J 88-5, School of ISE, Georgia Institute of Technology. 1988.
2. P. E. Gill and W. Murray, *A numerically stable form of the simplex algorithm*, Linear Algebra Appl. **7** (1973), 99–138.
3. ———, *Newton-type methods for linearly constrained optimization*, in P. E. Gill and W. Murray (eds), *Numerical Methods for Constrained Optimization*. Academic Press, London and New York 1974.
4. P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *A shur-complement method for sparse quadratic programming*, in *Reliable Numerical Computation*. M. G. Cox, and S. J. Hammarling, eds., Oxford University Press, Oxford, 1990, pp.113–138.
5. ———, *Inertia-controlling methods for general quadratic programming*, SIAM Rev. **33** (1991), 1–36.
6. D. Goldfarb and S. Liu, *An $O(n^3L)$ primal interior point algorithm for convex quadratic programming*, Math. Programming **49** (1991), 325–340.
7. N. I. M. Gould, *An algorithm for large-scale quadratic programming*, Report CSS 219, AERE Harwell, United Kingdom, 1989.
8. D. M. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill, 1972.
9. S. Kapoor and P. Vaidya, *Fast algorithms for convex quadratic programming and multicommodity flows*, Proceedings of the 18th annual ACM symposium on theory computing. 1986, 147–159.
10. M. Kojima, S. Mizuno and A. Yoshise, *A polynomial time algorithm for a class of linear complementarity problems*, Math. Programming **44** (1989), 1–26.
11. S. Mehrotra and J. Sun, *An algorithm for convex quadratic programming that requires $O(n^{3.5}L)$ arithmetic operations*, Math. Oper. Res. **15** (1990), 342–363.

12. R. C. Monteiro and I. Alder, *An $O(n^3L)$ primal-dual interior point algorithm for convex quadratic programming*, Math. Programming **44** (1989), 27–42.
13. A. B. Murtagh and M. A. Saunders, *MINOS 5.1 User's Guide*, Technical Report SOL 83-20R, Department of Operations Research, Stanford University, CA. 1983.
14. P. M. Pardalos and J. B. Rosen, *Constraint Global optimization : Algorithms and Applications*, Springer-Verlag, Berlin; New York. 1987.
15. D. B. Ponceleon, *Barrier methods for large-scale quadratic programming*, Technical Report SOL 91-2, Department of Operation Research, Stanford University, CA. 1991.
16. Y. Ye, *On affine scaling algorithms for nonconvex quadratic programming*, Math. Programming **56** (1992), 285–300.
17. Y. Ye and E. Tse, *An extension of Karmarkar's projective algorithm for convex quadratic programming*, Math. Programming **44** (1989), 157–179.

Department of Mathematics
Chungnam National University
Taejon 305-764, Korea