Chapter 3

QUADRATIC PROGRAMMING PROBLEMS

3.1 Introduction

Quadratic programming (QP) is family of methods, techniques and algorithms that can be used to minimize/maximize quadratic objective functions subject to linear constraints. An important branch of QP is convex QP where the objective function is a convex quadratic function. A classical method to solve convex QP is the Frank-Wolfe algorithm\cite{26} which can be considered as a direct extension of the well-known simplex method, which its complexity is exponential in the worst case\cite{2}. Therefore, there is a growing interest for developing efficient, robust and polynomial algorithms to solve the problem.
3.2 Statement of The Problem

The general quadratic program can be written as

\[
\text{Minimize } f(X) = \frac{1}{2}X'QX + C'X
\]

subject to \( AX \leq b \),

\[ X \geq 0. \tag{3.1} \]

Where \( Q = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nn} \end{pmatrix} \), \( A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \), \( X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \),

\[ b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}. \]

In matrix \( Q \), \( q_{jj} \) represents the coefficient of the term \( x_j^2 \) in the objective function and \( q_{ij} \) (where \( i \) is not equal to \( j \)) represents the coefficient of the term \( x_i x_j \) in the objective function. In matrix \( A \), \( a_{ij} \) represents the coefficient of \( x_j \) in the \( i^{th} \) constraint.

It is assumed that a feasible solution exists and that the constraint region is bounded. When the objective function \( f(X) \) is strictly convex for all feasible points the problem has a unique local minimum which is also the global minimum. A sufficient condition to guarantee strictly convexity is for \( Q \) to be positive definite.

How do we know that we have found the “optimum” for \( \text{minimize } f(X) \)?

Answer: Test the solution for the “necessary and sufficient conditions”. These conditions are known as Karush-Kuhn-Tucker (also known as Kuhn-Tucker) con-
KKT conditions are necessary condition for the local minimum solutions of problem (3.1) also they are sufficient for a global minimum when $Q$ is positive definite. Now we are going to obtain the KKT conditions for a local minimum of the quadratic program (3.1). Excluding the nonnegativity conditions, the Lagrangian function for the quadratic program (3.1) is

$$L(X, \mu) = C'X + 1/2X'QX + \mu(AX - b),$$

where $\mu$ is an $m$-dimensional row vector. The KKT conditions for a local minimum are given as follows.

1. $C' + X'Q + \mu A \geq 0$ (3.2a)
2. $AX - b \leq 0$ (3.2b)
3. $X'(C + QX + A'\mu') = 0$ (3.2c)
4. $\mu g(X) = \mu(AX - b) = 0$ (3.2d)
5. $X \geq 0$ (3.2e)
6. $\mu \geq 0$ (3.2f)

To put (3.2a)-(3.2f) into a more manageable form we introduce nonnegative surplus variables $Y \in \mathbb{R}^n$ to the inequalities in (3.2a) and nonnegative slack variables $V \in \mathbb{R}^m$ to the inequalities in (3.2b) to obtain the equations

$$C + QX + A'\mu' - Y = 0 \quad \text{and} \quad AX - b + V = 0$$
The KKT conditions can now be written with the constants moved to the right-hand-side.

\[ QX + A'\mu' - Y = -C \quad (3.3a) \]
\[ AX + V = b \quad (3.3b) \]
\[ X \geq 0, \quad \mu \geq 0, \quad Y \geq 0, \quad V \geq 0 \quad (3.3c) \]
\[ Y'X = 0, \quad \mu V = 0 \quad (3.3d) \]

The first two expressions are linear equalities, the third restricts all the variables to be nonnegative and the fourth prescribes complementary slackness.

### 3.2.2 Obtaining optimal solution of QPP using simplex algorithm

The simplex algorithm can be used to solve (3.3a)-(3.3d) by treating the complementary slackness conditions (3.3d) implicitly with a restricted basis entry rule. The procedure for setting up the linear programming model is as follows.

- Let the structural constraints be equations (3.3a) and (3.3b) defined by the KKT conditions.
- If any of the right-hand-side values are negative, multiply the corresponding equation by 1.
- Add an artificial variable to each equation.
- Let the objective function be the sum of the artificial variables.
- Put the resultant problem into simplex form.
The goal is to find the solution to the linear program that minimizes the sum of the artificial variables with the additional requirement that the complementarity slackness conditions be satisfied at each iteration. If the sum is zero, the solution will satisfy (3.3a)-(3.3d).

To satisfy (3.3d), the rule for selecting the entering variable must be modified with the following relationships.

\[ x_j \text{ and } y_j \text{ are complementary for } j = 1, \ldots, n. \]
\[ \mu_i \text{ and } v_i \text{ are complementary for } i = 1, \ldots, m. \]

The entering variable will be the one whose reduced cost is most negative provided that its complementary variable is not in the basis or would leave the basis on the same iteration. At the conclusion of the algorithm, the vector \( X \) defines the optimal solution and the vector \( \mu \) defines the optimal dual variables.

This approach has been shown to work well when the objective function is positive definite, and requires computational effort comparable to a linear programming problem with \( m + n \) constraints, where \( m \) is the number of constraints and \( n \) is the number of variables in the QP.

We solve the following problem using above procedure as an example.

\[
\begin{align*}
\min \; f(X) &= -8x_1 - 16x_2 + x_1^2 + 4x_2^2 \\
\text{subject to} & \quad x_1 + x_2 \leq 5, \\
& \quad x_1 \leq 3, \\
& \quad x_1, x_2 \geq 0.
\end{align*}
\]
Solution: The variable definitions are given below. As can be seen, the $Q$ matrix is positive definite so the KKT conditions are necessary and sufficient for a global minimum.

$$
C' = (-8 \quad -16), \quad Q = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad b' = (5, \quad 3), \\
X' = (x_1 \quad x_2), \quad Y' = (y_1 \quad y_2), \quad \mu = (\mu_1 \quad \mu_2) \quad \text{and} \quad V' = (v_1 \quad v_2).
$$

The linear constraints (3.3a) and (3.3b) take the following form.

$$
\begin{align*}
2x_1 + \mu_1 + \mu_2 - y_1 &= 8 \\
8x_2 + \mu_1 - y_2 &= 16 \\
x_1 + x_2 + v_1 &= 5 \\
x_1 + v_2 &= 3
\end{align*}
$$

To create the appropriate linear program, artificial variables are added to each constraint and minimize their sum.

Minimize $a_1 + a_2 + a_3 + a_4$

subject to

$$
\begin{align*}
2x_1 + \mu_1 + \mu_2 - y_1 + a_1 &= 8 \\
8x_2 + \mu_1 - y_2 + a_2 &= 16
\end{align*}
$$

all variables $\geq 0$ and complementarity conditions

Applying the modified simplex technique to this example, yields the sequence of iterations given in Table 3.1. The optimal solution to the original problem is $(x_1^*, x_2^*) = (3, 2)$.

Table 3.1: Simplex iterations for $QP$ example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Basic variables</th>
<th>Solution</th>
<th>Objective value</th>
<th>Entering variable</th>
<th>Leaving variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(a_1, a_2, a_3, a_4)$</td>
<td>(8, 16, 5, 3)</td>
<td>32</td>
<td>$x_2$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>2</td>
<td>$(a_1, x_3, a_3, a_4)$</td>
<td>(8, 2, 3, 3)</td>
<td>14</td>
<td>$x_1$</td>
<td>$a_3$</td>
</tr>
<tr>
<td>3</td>
<td>$(a_1, x_2, x_1, a_4)$</td>
<td>(2, 2, 3, 0)</td>
<td>2</td>
<td>$\mu_1$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>4</td>
<td>$(a_1, x_2, x_1, \mu_1)$</td>
<td>(2, 2, 3, 0)</td>
<td>2</td>
<td>$\mu_2$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>5</td>
<td>$(\mu_2, x_2, x_1, \mu_1)$</td>
<td>(2, 2, 3, 0)</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

57
3.3 Stochastic Search Algorithm

As mentioned in sections 1 and 2.2 a convex $QP$ problem is solved by simplex method and according to section 4.1 chapter 2, we need a new algorithm. Then, we propose a stochastic search algorithm which is useful when a near-optimal solution is acceptable and desired quickly.

We obtain the bounding box $B$ which contains the simplex set of feasible solution of $QP$ problem. All feasible solution of $QP$ problem are elements of $B$, though the converse may not hold.

Now, generate a random element from $B$ using Uniform distribution. If this element is feasible, then compute the objective function value at this point. Ignore all infeasible elements. Compute the smallest value of the objective function for these feasible solutions. In this process, a feasible solution is declared inadmissible if the value of the objective function is not smaller than the smallest value obtained till then. We keep track of the number of generated elements, the number of feasible solutions among these and the number of admissible points among the later. The procedure can be terminated after generating a specified number of random elements, feasible solutions or admissible solutions. The best solution at termination of the procedure is declared as the near-optimal solution.

3.3.1 Multiple starts

The simulation approach obtains a near-optimal solution for every starting value. Near-optimality of the final solution does not depend on the initial value and hence the rate of convergence can not be determined. Therefore, it may be convenient to begin with more than one initial value, generating an independent sequence of
solutions for every initial value. This will give us several end-points and the best of these will be better than the solution obtained from any one of them.

The maximum of the objective function value obtained through the above procedure is near-optimal. This method is not iterative in the sense that consecutive solutions may not improve the objective function. Also, it is simple from the mathematical point of view and there are no complicated computations.

Algorithmically, the method is described as follows:

1. Initialization in QP problem:

   Set $i = 1, z_{n-o} = 0, x_{n-o} = 0, i_o = 100$ (number of feasible points).

2. Generation in QP:

   - **Step 1** Generate $X_i \in B$ using uniform distribution on the bounding box $B$.
   - **Step 2** Test feasibility of $X_i$.
   - **Step 3** If $X_i$ is feasible, go to step 4. Otherwise, go to step 1.

3. P.3 Computation in QP:

   - **Step 4** Compute $z_i = f(X_i), i = 1, 2, \ldots, i_o$.
   - **Step 5** $z_{n-o} \leftarrow \max_i(z_i), i = 1, 2, \ldots, i_o$ and $x_{n-o} \leftarrow$ associated $x_i$ with $z_{n-o}$.

4. Termination in QP:

   - **Step 6** Keep the values of $X_{n-o}, z_{n-o}$ and stop.

Keep the smallest value of $z_{n-o}$ in all restarts then output values of $z_{n-o}$ and $X_{n-o}$.
3.3.2 Choice of probability distribution \((p_r)\)

For the probability distribution, we have two choices.

1. Uniform distribution.

   In this case, every element is generated from \(B\) with equal probability of selection. A significant limitation of uniform distribution is the progressively increasing rejection rate (rejection can occur for one of the two reasons: infeasibility and inadmissibility). This happens because points are generated with equal probability and rejection depends on the position of the current solution.

2. Non-Uniform distributions.

   Due to the limitation of uniform distribution, we propose to use a non-uniform distribution. First, we use the normal distribution with mean \(\mu_j = \text{upper bound of } x_j, j = 1, \ldots, n \) in case of less than or equal \((\leq)\) inequality constraints / lower bound of \(x_j, j = 1, \ldots, n\) in case of greater than or equal \((\geq)\) inequality constraints and unit variances. The results show that the normal distribution is better than uniform distribution in terms of objective function value. It means we can obtain better near-optimal solution. But in terms of number of infeasible points and total CPU time, uniform distribution is better.

   A major limitation of the normal distribution is that half of the generated points are expected to exceed the mean. This implies a rejection rate exceeding \(1/2\). This can be overcome by modifying the mean as follows.

   We take a fraction \(f \in (0, 1)\) and use the mean

   \[
   f \cdot \mu_j \quad 1 \leq j \leq n.
   \]
We tried some fractions to identify a good value for the fraction of mean and variance.

Another feature of the normal distribution is its unbounded support. Since the set of feasible solutions is bounded, we considered a distribution on a bounded support. In particular, we tried the multivariate Beta distribution, with the marginal distribution of $x_j$ on $B$ with parameters $\alpha, \beta > 0$ so that $\alpha < \beta$ or $\alpha = k \cdot \beta$ with $0 < k < 1$ because for an $QP$ problem the optimal solution is not necessarily an extreme point of the feasible region, even if the feasible region is a polyhedron. It may be on the bounding or interior the feasible region.

The gamma distribution is reasonable for the $QP$ problems with greater than or equal ($\geq$) because their feasible solution set is unbounded.

For a problem having integer optimal solution, a discrete probability distribution is found to be very efficient. This is more so if the coordinates of bounding points are small multiples of optimal $x_j$. 
3.4 Results of Simulation Studies

We consider some examples and implement the proposed stochastic search algorithm under some non-uniform distributions, namely Normal, Beta, Weibull and Gamma, with different parameters and two discrete distributions, namely Geometric and Poisson.

The following symbols are used in the tables.

- $Z_{\text{opt}}$ is the optimal objective function value.
- $Z_{\text{n-o}}$ is the near-optimal objective functions.
- # inf. P. is the number of infeasible points.
- # F.P. each It. is the number of feasible points in each iteration.
- IPBS is the number of infeasible points in iteration which gives the best solution.
- T is the total number of infeasible points in all iterations.
- CPU time Sec. is running time of stochastic search algorithm in terms of second.
Example 3.1:

Consider the following example.

\[\min f(X) = -8x_1 - 16x_2 + x_1^2 + 4x_2^2\]
subject to \[x_1 + x_2 \leq 5,\]
\[x_1 \leq 3,\]
\[x_j \geq 0.\]

The matrix form of problem is as follows:

\[\min f(X) = \frac{1}{2}X'QX + C'X\]
subject to \[AX \leq b,\]
\[X \geq 0.\]

Where \[Q = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix},\]
\[C = \begin{pmatrix} -8 \\ -16 \end{pmatrix},\]
\[A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\]
and \[b = \begin{pmatrix} 5 \\ 3 \end{pmatrix}.\]
Table 3.2: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering \textbf{Uniform} and \textbf{Normal} distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{\text{opt}}$</th>
<th>$Z_{n-o}$</th>
<th>#inf. P.</th>
<th># F. P. each It.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Uniform}(0, \mu_j)$</td>
<td>-31</td>
<td>-30.84</td>
<td>42</td>
<td>440</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\text{Uniform}(0, \mu_j)$</td>
<td>-31</td>
<td>-30.96</td>
<td>44</td>
<td>2236</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>$N(\mu_j, 1)$</td>
<td>-31</td>
<td>-30.92</td>
<td>5893</td>
<td>64549</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$N(1/2 \mu_j, 1)$</td>
<td>-31</td>
<td>-30.9</td>
<td>44</td>
<td>497</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$N(1/2 \mu_j, \mu_j^2)$</td>
<td>-31</td>
<td>-30.85</td>
<td>839</td>
<td>8847</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$N(1/2 \mu_j, 1/2 \mu_j^2)$</td>
<td>-31</td>
<td>-30.94</td>
<td>471</td>
<td>4341</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 3.3: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering \textbf{Beta} distribution.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{\text{opt}}$</th>
<th>$Z_{n-o}$</th>
<th>#inf. P.</th>
<th># F. P. each It.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_j * \text{Beta}(0.5, 0.75)$</td>
<td>-31</td>
<td>-30.95</td>
<td>14</td>
<td>241</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * \text{Beta}(0.75, 0.5)$</td>
<td>-31</td>
<td>-30.9991</td>
<td>81</td>
<td>953</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * \text{Beta}(0.25, 0.05)$</td>
<td>-31</td>
<td>-31</td>
<td>534</td>
<td>4715</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * \text{Beta}(0.5, 0.75)$</td>
<td>-31</td>
<td>-30.996</td>
<td>29</td>
<td>1236</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * \text{Beta}(0.75, 0.5)$</td>
<td>-31</td>
<td>-30.997</td>
<td>111</td>
<td>4785</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * \text{Beta}(0.25, 0.05)$</td>
<td>-31</td>
<td>-31</td>
<td>451</td>
<td>22837</td>
<td>100</td>
<td>50</td>
</tr>
</tbody>
</table>
Table 3.4: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering Gamma distribution.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{opt}$</th>
<th>$Z_{n-o}$</th>
<th>#inf. P.</th>
<th># F. P. each It.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_j \ast \text{Gamma}(1, 2)$</td>
<td>$-31$</td>
<td>$-30.96$</td>
<td>676</td>
<td>7679</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast \text{Gamma}(1, 1)$</td>
<td>$-31$</td>
<td>$-30.92$</td>
<td>209</td>
<td>2020</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast \text{Gamma}(1, 0.5)$</td>
<td>$-31$</td>
<td>$-30.75$</td>
<td>53</td>
<td>501</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast \text{Gamma}(0.5, 0.5)$</td>
<td>$-31$</td>
<td>$-30.57$</td>
<td>9</td>
<td>105</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast \text{Gamma}(1, 2)$</td>
<td>$-31$</td>
<td>$-30.974$</td>
<td>690</td>
<td>37483</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j \ast \text{Gamma}(1, 1)$</td>
<td>$-31$</td>
<td>$-30.95$</td>
<td>231</td>
<td>10165</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j \ast \text{Gamma}(1, 0.5)$</td>
<td>$-31$</td>
<td>$-30.83$</td>
<td>42</td>
<td>2420</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j \ast \text{Gamma}(0.5, 0.5)$</td>
<td>$-31$</td>
<td>$-30.88$</td>
<td>17</td>
<td>599</td>
<td>100</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3.5: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering Piosson and Geometric distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{opt}$</th>
<th>$Z_{n-o}$</th>
<th>#inf. P.</th>
<th># F. P. each It.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piosson($\mu_j$)</td>
<td>$-31$</td>
<td>$-31$</td>
<td>391</td>
<td>391</td>
<td>1</td>
<td>0.22</td>
</tr>
<tr>
<td>Piosson($1/2 \mu_j$)</td>
<td>$-31$</td>
<td>$-31$</td>
<td>32</td>
<td>32</td>
<td>1</td>
<td>0.15</td>
</tr>
<tr>
<td>Geometric(0.25)</td>
<td>$-31$</td>
<td>$-31$</td>
<td>32</td>
<td>32</td>
<td>1</td>
<td>0.12</td>
</tr>
<tr>
<td>Geometric(0.5)</td>
<td>$-31$</td>
<td>$-31$</td>
<td>12</td>
<td>12</td>
<td>1</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Example 3.2:

Consider the following problem.

\[ \text{Min } f(X) = -x_1 - x_2 + x_5 + 3/2x_1^2 + 2x_2^2 + 2x_3^2 + 4x_4^2 + 2x_5^2 - x_1x_2 + x_1x_3 \]

subject to

\[
\begin{align*}
  x_1 - x_2 - 3x_3 - x_4 + 2x_5 & \leq 6, \\
  -x_1 - x_2 + 2x_3 + x_4 - 2x_5 & \leq 2, \\
  -7x_2 - 2x_3 - 2x_4 - 5x_5 & \leq 3, \\
  -x_1 + 5x_2 + x_3 - 3x_4 - x_5 & \leq 6, \\
  -x_1 - x_3 + x_4 - 4x_5 & \leq 4,
\end{align*}
\]

\[ x_1, x_2, x_3, x_4, x_5 \geq 0. \]

The Stochastic search algorithm under different probability distributions on this example was implemented and 100 feasible solutions are generated in each iteration. The obtained results are in the following tables. For this example Uniform and Normal distributions are not suitable because the optimal solution is not located close to the upper bounds of \(x_j\)s.
Table 3.6: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering Beta distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{opt}$</th>
<th>$Z_{n-opt}$</th>
<th>#inf. P.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_j * Beta(0.75, 0.5)$</td>
<td>-0.4091</td>
<td>0.19</td>
<td>29</td>
<td>287</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.5, 0.75)$</td>
<td>-0.4091</td>
<td>-0.016</td>
<td>11</td>
<td>177</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.05, 0.25)$</td>
<td>-0.4091</td>
<td>-0.34</td>
<td>17</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.75, 0.5)$</td>
<td>-0.4091</td>
<td>-0.19</td>
<td>20</td>
<td>1487</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.5, 0.75)$</td>
<td>-0.4091</td>
<td>-0.34</td>
<td>15</td>
<td>758</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.05, 0.25)$</td>
<td>-0.4091</td>
<td>-0.408</td>
<td>14</td>
<td>493</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3.7: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering Gamma distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{opt}$</th>
<th>$Z_{n-opt}$</th>
<th>#inf. P.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_j * Gamma(1, 1)$</td>
<td>-0.4091</td>
<td>0.42</td>
<td>112</td>
<td>1160</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.5, 1)$</td>
<td>-0.4091</td>
<td>-0.2616</td>
<td>67</td>
<td>621</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.05, 1)$</td>
<td>-0.4091</td>
<td>-0.376</td>
<td>6</td>
<td>48</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(1, 1)$</td>
<td>-0.4091</td>
<td>0.18</td>
<td>106</td>
<td>5968</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.5, 1)$</td>
<td>-0.4091</td>
<td>-0.375</td>
<td>68</td>
<td>3110</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.05, 1)$</td>
<td>-0.4091</td>
<td>-0.408</td>
<td>2</td>
<td>251</td>
<td>50</td>
</tr>
</tbody>
</table>
Example 3.3:

Consider the following problem.

\[ Min \ f(X) = 4x_1 + 3x_2 + x_3 - x_4 + x_1^2 + 2x_2^2 + 3x_3^2 + 2x_4^2 + x_5^2 - x_1x_2 + x_3x_4 + 2x_4x_5 \]

subject to

\[
\begin{align*}
    x_1 + x_2 - x_3 + x_5 & \leq 6, \\
    x_2 + 2x_3 - x_4 & \leq 8, \\
    x_1 + 4x_3 - x_4 - x_5 & \leq 5, \\
    x_1 + x_2 + 2x_4 & \leq 3,
\end{align*}
\]

\[ x_1, x_2, x_3, x_4, x_5 \geq 0. \]

The Stochastic search algorithm under different probability distributions on this example was implemented and 100 feasible solutions are generated in each iteration. The obtained results are in the following tables. For this example Uniform and Normal distributions are not suitable because the optimal solution is not located close to the upper bounds of \( x_j \)s.
Table 3.8: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering Beta distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{opt}$</th>
<th>$Z_{n-o}$</th>
<th>$#$ inf. P.</th>
<th>$#$ Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_j * Beta(0.75, 0.5)$</td>
<td>-0.125</td>
<td>1.63</td>
<td>1612</td>
<td>15847</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.5, 0.75)$</td>
<td>-0.125</td>
<td>1.019</td>
<td>212</td>
<td>2160</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.05, 0.25)$</td>
<td>-0.125</td>
<td>-0.1248</td>
<td>47</td>
<td>405</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.75, 0.5)$</td>
<td>-0.125</td>
<td>1.12</td>
<td>1564</td>
<td>72268</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.5, 0.75)$</td>
<td>-0.125</td>
<td>0.081</td>
<td>217</td>
<td>11369</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.05, 0.25)$</td>
<td>-0.125</td>
<td>-0.125</td>
<td>40</td>
<td>2062</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3.9: Near-Optimal values of objective function in QP problem by using proposed stochastic search algorithm with considering Gamma distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{opt}$</th>
<th>$Z_{n-o}$</th>
<th>$#$ inf. P.</th>
<th>$#$ Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_j * Gamma(1, 1)$</td>
<td>-0.125</td>
<td>1.22</td>
<td>3103</td>
<td>30129</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.5, 1)$</td>
<td>-0.125</td>
<td>0.03</td>
<td>288</td>
<td>2468</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.05, 1)$</td>
<td>-0.125</td>
<td>-0.1246</td>
<td>10</td>
<td>66</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(1, 1)$</td>
<td>-0.125</td>
<td>1.056</td>
<td>3732</td>
<td>154866</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.5, 1)$</td>
<td>-0.125</td>
<td>-0.07</td>
<td>212</td>
<td>12455</td>
<td>50</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.05, 1)$</td>
<td>-0.125</td>
<td>-0.125</td>
<td>8</td>
<td>363</td>
<td>50</td>
</tr>
</tbody>
</table>

69
3.5 Conclusions

In this paper, we proposed a stochastic search algorithm which is an interior point method for finding the near-optimal solution for Quadratic Programming problem with positive (semi) definite $Q$ matrix subject to linear inequality ($\leq$) constraints. The proposed algorithm uses multiple starts to find the best near-optimal solution which is global because of convexity property of problem. The proposed algorithm is easy in programming and does not need to derivative mathematics knowledge. It is not an iterative procedure. Gamma and Beta probability distributions are reasonable distribution to generate feasible points from feasible region of considered problem also, discrete distributions like Geometry and Poisson are very efficient in certain special cases.

All this work has been submitted to an international journal for publication.