Chapter 4

GENERAL NON-LINEAR PROGRAMMING PROBLEMS

4.1 Introduction

In this chapter we consider a general nonlinear programming problem. Nonlinear programming problem is a direct extension of linear programming where we replace linear model functions by nonlinear ones. In many real-life problems, the objective function may be nonlinear but the set of constraints may be linear or nonlinear and the most general class of optimization problems is the class of problems where both the objective function and the constraints are nonlinear. These problems can be solved by using a variety of methods such as penalty and barrier methods, gradient projection methods and sequential quadratic programming (SQP) methods [25]. Nonlinear programming is “hard”, because there does not exist an algorithm that can solve every NLP problem efficiently in practice then, we apply the stochastic search algorithm to find the near-optimal solution of general nonlinear programming problem.
4.2 Problem Statement and Basic Definitions

The general form of nonlinear programming problem is in form of following [69]:

\[ \begin{align*}
\text{Minimize} & \quad f(X) \quad X \in \mathbb{R}^n \\
\text{Subject to} & \quad g_i(X) \leq 0 \quad , i = 1, 2, \cdots, m \\
& \quad h_i(X) = 0 \quad , i = 1, \cdots, l.
\end{align*} \tag{4.1} \]

Here, \( X \) is an \( n \) dimensional vector of decision variables, \( f(X) \) is the objective function to be minimized under inequality and equality constraints given by \( g(X) \) and \( h(X) \), respectively. It is assumed that these functions are continuously differentiable in \( \mathbb{R}^n \). Vector \( X \) satisfying all the constraints is called a feasible solution to the problem. The collection of all such solutions forms the feasible region. The nonlinear programming problem is to find a feasible point \( X^* \) such that \( f(X^*) \leq f(X) \) for each feasible point \( X \). Such a point \( X^* \) is called an optimal solution to the problem. A nonlinear programming can be stated as maximization problem, and the inequality constraints can be written in the form \( g_i(X) \leq 0 \) for \( i = 1, 2, \cdots, m \).

To illustrate, consider the following problem:

\[ \begin{align*}
\text{Minimize} & \quad f(X) = (x_1 - 3)^2 + (x_2 - 2)^2 \\
\text{subject to} & \quad x_1^2 - x_2 - 3 \leq 0, \\
& \quad x_2 - 1 \leq 0, \\
& \quad X \geq 0.
\end{align*} \]

Figure 4.1 illustrates the feasible region. The problem is to find the point in the feasible region with minimum possible \( (x_1 - 3)^2 + (x_2 - 2)^2 \). Note that points \((x_1, x_2)\) with \( (x_1 - 3)^2 + (x_2 - 2)^2 = c \) represent a circle with radius \( \sqrt{c} \) and center
(3,2). Since we wish to minimize $c$, we must find the circle with the smallest radius that intersects the feasible region. As shown in Figure 4.1, the smallest such circle has $c = 2$ and intersects the feasible region at the point (2,1) and the optimal value of objective function is equal to 2.

![Figure 4.1 Geometric solution of a nonlinear problem](image)

Obviously, the above approach of solving the problem geometrically is suitable only for small problem and not practical for problems with more than two variables or for problems with complicated objective and constraint functions.

Here, we are introducing some method which are using to solve constraint nonlinear programming problems.
4.3 Penalty and Barrier Methods

In nonlinear optimization penalty and barrier methods are normally used to solve constrained problems. The basic principle of these methods is transforming the constrained problem to an unconstrained problem by adding a penalty term to the objective function. This term penalizes, i.e. increases the value of the objective function when the constraints are violated or even when the bounding is approached. Adding penalty is for infeasibility and forcing the solution to feasibility and subsequent optimum and adding a barrier is for insuring that a feasible solution never becomes infeasible. Hence, we can call the penalty method as an exterior penalty and the barrier method as interior penalty method.

4.3.1 Penalty function method

Penalty functions have been a part of the literature on constrained optimization for decades [48]. In general, a penalty function approach is as follows. Consider the constrained optimization problem (4.1). Whose feasible region we denote by

\[ \Omega = \{ X \in \mathbb{R}^n \mid g_i(X) \leq 0 \quad i = 1, \ldots, m, \quad h_i(X) = 0 \quad i = 1, \ldots, l \}. \]

Penalty methods are designed to solve (4.1) by solving a sequence of specially constructed unconstrained optimization problems [48]. The feasible region of equation (4.1) is expanded from \( \Omega \) to all of \( \mathbb{R}^n \), but a penalty is added to the objective function for points that lie outside of the original feasible region \( \Omega \).

**Definition 4.1** A function \( C(X) : \mathbb{R}^n \rightarrow \mathbb{R} \) is called a penalty function for equation (4.1) if \( C(X) \) satisfies: \( \{ C(X) = 0 \ if \ g(X) \leq 0 \ h(X) = 0 \} \) and \( \{ C(X) > 0 \ if \ g(X) > 0 \ or \ h(X) \neq 0 \} \)[6].
Penalty functions are typically defined by:

\[ C(X) = \sum_{i=1}^{m} \phi(g_i(X)) + \sum_{i=1}^{l} \psi(h_i(X)) \]  

(4.2)

Where \( \{\phi(g_i(X)) = 0 \text{ if } g_i(X) \leq 0\} \) and \( \{\phi(g_i(X)) > 0 \text{ if } g_i(X) > 0\} \), \( \{\psi(h_i(X)) = 0 \text{ if } h_i(X) = 0\} \) and \( \{\psi(h_i(X)) > 0 \text{ if } h_i(X) \neq 0\} \).

In theory, more general functions satisfying the definition can conceptually be used. We then consider solving the following penalty program:

\[ \min(f(X) + \mu C(X)) \]  

(4.3)

where \( X \in \mathbb{R}^n \) for an increasing sequence of constants \( \mu \) as \( \mu \to \infty \). In problem (4.3), we are assigning a penalty to the violated constraints. The scalar quantity \( \mu \) is called the penalty parameter. Let \( \{\mu_k\}_{k=1}^{\infty} \) be a increasing sequence of penalty parameters that satisfies \( \mu_{k+1} > \mu_k \) for \( \forall \ k \) and \( \lim_{k \to \infty}(\mu_k) \to +\infty \). Let \( F(X) = f(X) + \mu C(X) \) and let \( X^k \) be the exact solution to the problem (4.3), and let \( X^* \) denote any optional solution of (4.1). The following Lemma presents some basic properties of penalty methods.

**Lemma 4.1** Properties of penalty methods [6]

1. \( F(\mu_k, X^k) \leq F(\mu_{k+1}, X^{k+1}) \)
2. \( C(X^k) \geq C(X^{k+1}) \)
3. \( f(X^k) \leq f(X^{k+1}) \)
4. \( f(X^*) \geq F(\mu_k, X^k) \geq f(X^k) \)

An often used class of penalty functions is:

\[ C(X) = \sum_{i=1}^{m_1} \max[0, g_i(X)]^c + \sum_{i=1}^{m_2} |h_i(x)|^c \text{ where } c \geq 1 \quad m = m_1 + m_2 \]  

(4.4)
if $c = 1$, $C(X)$ in equation (4.4) is called the linear penalty function. This function may not be differentiable at points where $g_i(X) = 0$ or $h_i(X) = 0$ for some $i$. Setting $c = 2$ is the most common form of (4.4) that is used in practice, and is called the quadratic penalty function.

### 4.3.2 Computational difficulties of penalty functions

The solution of the penalty problem can be made arbitrary close to the optimal solution of the original problem by choosing $\mu$ sufficiently large. Choosing a very large $\mu$ for solving the penalty problem causes some computational difficulties. with a large $\mu$ (more emphasis is placed on feasibility) and most procedures for unconstrained optimization will move quickly toward a feasible point. Even though this point may be far from the optimal and termination could early occur [6].

### 4.3.3 Karush-Kuhn-Tucker multipliers

Suppose the penalty function $C(X)$ is defined as (4.2). $C(X)$ might not be continuously differentiable, since the functions $g_i(X)$ are not differentiable at points $X$ where $g_i(X) = 0$. However, if we assume that the functions $\phi(y)$ and $\varphi(y)$ are continuously differentiable and $\phi'(0) = 0$ then $C(X)$ is differentiable whenever the functions $g(X)$ and $h(X)$ are differentiable, and we can write:

$$\nabla C(X) = \sum_{i=1}^{m} \phi'(g_i(X)) \nabla g_i(X) + \sum_{i=1}^{l} \varphi'(h_i(X)) \nabla h_i(X) \quad (4.5)$$

Let $X^k$ solve (4.3). Then $X^k$ will satisfy:

$$\nabla f(X^k) + \mu_k \nabla C(X^k) = 0 \quad (4.6)$$

that is

$$\nabla f(X^k) + \mu_k \left[ \sum_{i=1}^{m} \phi'(g_i(X^k)) \nabla g_i(X^k) + \sum_{i=1}^{l} \varphi'(h_i(X^k)) \nabla h_i(X^k) \right] = 0 \quad (4.7)$$
The $u^k_i$ and $v^k_i$ are called Karush-Kuhn-Tucker multipliers.

**Lemma 4.2** Suppose $\phi(y)$ and $\varphi(y)$ are continuously differentiable and satisfy $\phi(0) = 0$, and that $f(X)$, $g(X)$, and $h(X)$ are differentiable. Let $u^k_i$, $v^k_i$ be defined by equation (4.8). Then if $X^k \to \bar{X}$ and $\bar{X}$ satisfies the linear independence condition for gradient vectors of active constraints, then $u^k_i \to \bar{u}$ and $v^k_i \to \bar{v}$ where $\bar{u}$ and $\bar{v}$ are a vector of Karush-Kuhn-Tucker multipliers for the optimal solution $\bar{X}$ of (4.1) [6].

### 4.3.4 Barrier function methods

Similar to penalty functions, barrier functions are also used to convert a constrained problem into an unconstrained problem or into a sequence of unconstrained problems. These functions set a barrier against leaving the feasible region. In fact, the idea in a barrier method is to prevent points $X$ from approaching the bounding of the feasible region [30].

If the optimal solution occurs at the bounding of feasible region, the procedure moves from the interior to the bounding [6]. Also in this method since convergence is from the inside of feasible region we can call this method as *Interior penalty method*. 

\begin{align*}
\nabla f(X^k) + \sum_{i=1}^{m} u^k_i \nabla g_i(X^k) + \sum_{i=1}^{l} v^k_i \nabla h_i(X^k) &= 0 \quad (4.9)
\end{align*}
**Definition 4.2** A barrier function for (4.1) is any function $B(X) : \mathbb{R}^n \to \mathbb{R}$ that satisfies, $B(X) = 0$ for all $X$ that satisfy $g(X) < 0$, and $B(X) \to \infty$ as $\lim_{x \to 0 \max_i g_i(X)} = 0$.

Define barrier minimization problem:

$$\min \ (f(X) + \mu B(X))$$
$$s.t. \ \ g(X) < 0, \ for \ X \in \mathbb{R}^n \quad (4.10)$$

for a sequence of $\mu_k \to \infty$. The following Lemma presents some basic properties of barrier methods.

**Lemma 4.3** Let $F(X, \mu) = f(X) + \mu g(X)$. Let the sequence $\{\mu_k\}$ satisfy $\mu_{k+1} > \mu_k$, $\mu_k \to \infty$ as $k \to \infty$. Let $X^k$ denote the exact solution to (4.10).

- $F(\mu_k, X^k) \geq F(\mu_{k+1}, X^{k+1})$
- $B(X^k) \leq B(X^{k+1})$
- $f(X^k) \geq f(X^{k+1})$
- $f(X^*) \leq f(X^k) \leq F(\mu_k, X^k)$

**Theorem 4.1** Suppose $f(X)$, $g(X)$ and $B(X)$ are continuous functions. Let $X^k$, $k = 1, 2 \cdots$ be a sequence of solutions of $B(\mu_k)$. Suppose there exists an optimal solution $X^*$ of (4.1) for which $N(\epsilon, X^*) \cap \{X | g(X) < 0\} \neq \emptyset$ for every $\epsilon > 0$. Then any limit point $\bar{X}$ of $\{X^k\}$ solves (4.1) [6].

78
4.3.5 Computational difficulties of barrier functions

The use of barrier functions for solving constrained nonlinear programming problems faces several computational difficulties. First, the search must start with a feasible point. For some problems, finding such a point is not an easy task. Also, because of the structure of the barrier function $B$, and for small values of the parameter $\mu$, most search techniques may face serious ill-conditioning and round-off difficulties while solving the problem to minimize $f(X) + \mu B(X)$ over $X \in \mathbb{R}^n$, specially as the bounding of the region $\{X : g(X) \leq 0\}$ is approached and an explicit check of the value of the constraint function $g$ is needed to guarantee that we do not leave the feasible region.
4.4 Sequential Quadratic Programming Methods

Among the methods which are used for solving general constrained problems, sequential quadratic programming (SQP) algorithms have proved highly effective for solving general constrained problems with smooth objective and constraint functions [29, 11]. SQP methods attempt to solve a constrained nonlinear programming problem directly rather than convert it to a sequence of unconstrained minimization problems. Whereas, Penalty and barrier methods are indirect ways of solving constrained optimization problems.

SQP methods for the general nonlinear constrained optimization problem (4.1) were first studied by Wilson [64]. A recent survey of SQP algorithms can be found in [11]. As the name implies, sequential quadratic programming (SQP) methods are iterative methods which solves at each iteration a quadratic programming problem (QP). For making simple this introduction, we begin by discussing the SQP for equality constrained nonlinear programming.

4.4.1 SQP problems with equality constraints

Consider the optimization problem

\[
\text{Minimize} \quad f(X) \quad X \in \mathbb{R}^n \\
\text{subject to} \quad h_i(X) = i = 1, \ldots, l
\]  

(4.11)

Where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( h_i : \mathbb{R}^n \rightarrow \mathbb{R} \) are smooth functions. We assume that feasible region \( \Omega \) described by (4.11) is nonempty and that \( l \leq n \). We know that the first-order necessary conditions for \( X^* \) to be a local minimizer of the problem
(4.11) are that there exists a $\lambda^* \in \mathbb{R}^l$ such that
\[ \nabla \mathcal{L}(X^*, \lambda^*) = 0 \] (4.12)
where $\mathcal{L}(X, \lambda)$ is the Lagrangian defined by
\[ \mathcal{L}(X, \lambda) = f(X) - \sum_{i=1}^l \lambda_i h_i(X) \]
and the gradient operation in (4.12) is performed with respect to $X$ and $\lambda$, i.e.
\[ \lambda = \begin{bmatrix} \nabla_x \\ \nabla_{\lambda} \end{bmatrix} \]

If set $\{X_k, \lambda_k\}$ is the $k$th iterate, which is assumed to be sufficiently close to $\{X^*, \lambda^*\}$, i.e. $X_k \approx X^*$ and $\lambda_k \approx \lambda^*$, we need to find an increment $\{\delta_x, \delta_\lambda\}$ such that the next iterate $\{X_{k+1}, \lambda_{k+1}\} = \{X_k + \delta_x, \lambda_k + \delta_\lambda\}$ is closer to $\{X^*, \lambda^*\}$. If we approximate $\nabla \mathcal{L}(X_{k+1}, \lambda_{k+1})$ by using the first two terms of the Taylor series of $\nabla \mathcal{L}$ for $\{X_k, \lambda_k\}$, i.e.
\[ \nabla \mathcal{L}(X_{k+1}, \lambda_{k+1}) \approx \nabla \mathcal{L}(X_k, \lambda_k) + \nabla^2 \mathcal{L}(X_k, \lambda_k) \begin{bmatrix} \delta_x \\ \delta_\lambda \end{bmatrix} \]
then $\{X_{k+1}, \lambda_{k+1}\}$ is an approximation of $\{X^*, \lambda^*\}$ if the increment $\{\delta_x, \delta_\lambda\}$ satisfies the equality
\[ \nabla^2 \mathcal{L}(X_k, \lambda_k) = -\nabla \mathcal{L}(X_k, \lambda_k) \] (4.13)

More specifically, we can write (4.13) in term of the Hessian of the Lagrangian, $W$, for $(X, \lambda) = (X_k, \lambda_k)$ and the Jacobian, $H$ for $X = X_k$ as
\[ \begin{bmatrix} W_k & -H_k^T \\ -H_k^T & 0 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_\lambda \end{bmatrix} = \begin{bmatrix} H_k^T \lambda_k - g_k \\ h_k \end{bmatrix} \] (4.14)
where
\[ W_k = \nabla^2_x f(X_k) - \sum_{i=1}^l (\lambda_k)_i \nabla^2_x h_i(X_k) \] (4.15)
\[ H_k = \begin{bmatrix} \nabla_T x h_1(X_k) \\ \nabla_T x h_2(X_k) \\ \vdots \\ \nabla_T x h_p(X_k) \end{bmatrix} \]  
(4.16)

\[ f_k = \nabla_x f(X_k) \]  
(4.17)

\[ h_k = \begin{bmatrix} h_1(X_k) \\ h_2(X_k) \\ \cdots \\ h_l(X_k) \end{bmatrix}^T \]  
(4.18)

If \( W_k \) is positive definite and \( H_k \) has full row rank, then the matrix at the left-hand-side of (4.14) is nonsingular and symmetric and the system of equations (4.14) can be solved efficiently for \( \{ \delta x, \delta \lambda \} \) as shown in Chapt. 4 of [31]. Equation (4.14) can also be written as

\[ W_k \delta x + f_k = H_k^T \lambda_{k+1} \]  
(4.19)

\[ H_k \delta x = -h_k \]  
(4.20)

and these equations may be interpreted as the first-order necessary conditions for \( \delta x \) to be a local minimizer of the QP problem

\[
\text{Minimize } \frac{1}{2} \delta^T W_k \delta + \delta^T f_k
\]  
(4.21)

\[
\text{Subject to: } H_k \delta = h_k
\]  
(4.22)

If \( W_k \) is positive definite and \( H_k \) has full row rank, the minimizer of the problem in (4.21)-(4.22) can be found by using Quadratic programming method. Once the minimizer, \( \delta x \) is obtained the next iterate is set to \( X_{k+1} = X_k + \delta x \) and the Lagrange multiplier vector \( \lambda_{k+1} \) is determined as

\[ \lambda_{k+1} = (H_k H_k^T)^{-1} H_k (W_k \delta x + f_k) \]  
(4.23)
by using (4.19). With $X_{k+1}$ and $\lambda_{k+1}$ known, $W_{k+1}$, $g_{k+1}$, $H_{k+1}$ and $h_{k+1}$ can be evaluated. The iterations are continued until $\|\delta_x\|$ is sufficiently small to terminate the algorithm. We see that the entire solution procedure consists of solving a series of QP subproblems in a sequential manner and, as a consequence, the method is often referred to as the sequential quadratic programming (SQP) method.

Algorithmically, the SQP method is described as follows.

Algorithm 4.1.

• **Step 1** Set $\{x, \lambda\} = \{X_0, \lambda_0\}$, $k = 0$ and initialize the tolerance $\epsilon$.

• **Step 2** Evaluate $W_k$, $H_k$, $f_k$ and $h_k$ using (4.15)-(4.18).

• **Step 3** Solve the QP problem in (4.21)-(4.22) for $\delta$ and compute Lagrange multiplier $\lambda_{k+1}$ using (4.23).

• **Step 4** Set $X_{k+1} = X_k + \delta_x$. If $\|\delta_x\| \leq \epsilon$, output $X^* = X_{k+1}$ and stop. Otherwise, set $k = k + 1$ and go to Step 2.

To apply above algorithm we consider following example.

**Example 4.1**

Minimize $f(X) = -x_1^4 - 2x_2^4 - x_3^4 - (x_1x_2)^2 - (x_1x_3)^2$

subject to:

$x_1^4 + x_2^4 + x_3^4 - 25 = 0$

$8x_1^2 + 14x_2^2 + 7x_3^2 - 56 = 0$

with $X_k = [x_1 \ x_2]^T$ and $\lambda_k = [\lambda_1 \ \lambda_2]^T$, Step 2 of algorithm 4.1 gives

$$W_k = \begin{bmatrix}
-12x_1^2 - 2x_2^2 - 2x_3^2 & -4x_1x_2 & -4x_1x_3 \\
-12\lambda_1x_1^2 - 16\lambda_2 & -24x_1^2 - 2x_3^2 & 0 \\
-4x_1x_2 & -12\lambda_1x_2^2 - 28\lambda_2 & 0 \\
-4x_1x_3 & 0 & -12x_2^2 - 2x_1^2 \\
-4x_1x_3 & 0 & -12\lambda_1x_3^2 - 14\lambda_2
\end{bmatrix}$$
\[
\begin{align*}
f_k &= \begin{bmatrix} -4x_1^3 - 2x_1x_2^2 - 2x_1x_3^2 \\ -8x_2^3 - 2x_1^2x_2 \\ -4x_3^3 - 2x_1^2x_3 \end{bmatrix} \\
H_k &= \begin{bmatrix} 4x_1^3 & 4x_2^3 & 4x_3^3 \\ 16x_1 & 28x_2 & 14x_3 \end{bmatrix} \\
h_k &= \begin{bmatrix} x_1^4 + x_2^4 + x_3^4 - 25 \\ 8x_1^2 + 14x_2^2 + 7x_3^2 - 56 \end{bmatrix}
\end{align*}
\]

with \( x_0 = [3 \ 1.5 \ 3]^T, \ \lambda_0 = [-1 \ -1]^T \) and \( \epsilon = 10^{-8} \), using the algorithm 4.1 after 10 iterations the algorithm converges to

\[
X^* = \begin{bmatrix} 1.874065 \\ 0.465820 \\ 1.884720 \end{bmatrix}, \quad \lambda^* = \begin{bmatrix} -1.223464 \\ -0.274937 \end{bmatrix}
\]

and \( f(X^*) = -38.384828 \). To examine the \( X^* \) is a local minimizer, we compute the Jacobian of the constraints \( H \) at \( X^* \) and find the null space of \( H^T (X^*) \). The null space of \( H^T (X^*) \) is

\[
N(X^*) = \begin{bmatrix} -0.696840 \\ 0.222861 \\ 0.681724 \end{bmatrix}
\]

This leads to

\[
N^T (X^*) \nabla^2_\lambda \mathcal{L} (X^*, \lambda^*) N (X^*) = 20.4 > 0
\]

Then, \( X^* \) is a local minimizer of the problem [4, 24].
4.4.2 SQP problems with inequality constraints

The SQP method of equality constraints can be extended to the case of inequality constraints. Let us consider the following problem

\[
\begin{align*}
\text{Minimize} & \quad f(X) \quad X \in \mathbb{R}^n \\
\text{Subject to:} & \quad g_i(X) \leq 0 \quad i = 1, \ldots, m
\end{align*}
\] (4.24)

where \(f(X)\) and \(g_i(X)\) are smooth functions and the feasible region \(\Omega\) described by (4.24) is nonempty. According to SQP method for equality constraints, we need to find an increment \(\{\delta_x, \delta_\mu\}\) for the \(k\)th iterate \(\{X_k, \mu_k\}\) approximate the Karush-Kuhn-Tucker (KKT) conditions which are following [4]

\[
\nabla_x \mathcal{L}(X, \lambda) = 0
\]
\[
g_i(X) \leq 0 \quad \text{for } i = 1, 2, \ldots, m
\]
\[
\mu \geq 0
\]
\[
\mu_i g_i(X) = 0 \quad \text{for } i = 1, 2, \ldots, m
\]

in the sense that

\[
\nabla_x \mathcal{L}(X_{k+1}, \mu_{k+1}) \approx \nabla_x \mathcal{L}(X_k, \mu_k) + \nabla^2_x \mathcal{L}(X_k, \mu_k) \delta_x + \nabla^2_{x \mu} \mathcal{L}(X_k, \mu_k) \delta_\mu = 0
\] (4.25)

\[
g_i(X_k, \delta_x) \approx g_i(X_k) + \delta_x^T \nabla_x g_i(X_k) \leq 0 \quad \text{for } i = 1, \ldots, m
\] (4.26)

\[
\mu_{k+1} \geq 0
\] (4.27)

and

\[
[g_i(X_k) + \delta_x^T \nabla_x g_i(X_k)] (\mu_{k+1})_i = 0 \quad \text{for } i = 1, \ldots, m
\] (4.28)
The Lagrangian $\mathcal{L}(X, \mu)$ in this case is defined as

$$
\mathcal{L}(X, \mu) = f(X) - \sum_{i=1}^{m} \mu_i g_i(X)
$$

(4.29)

Hence

$$
\nabla_x \mathcal{L}(X_k, \mu_k) = \nabla_x f(X_k) - \sum_{i=1}^{m} (\mu_k)_i \nabla_x g_i(X_k) = f_k - G^T_k \mu_k
$$

$$
\nabla^2_x \mathcal{L}(X_k, \mu_k) = \nabla^2_x f(X_k) - \sum_{i=1}^{m} (\mu_k)_i \nabla^2_x g_i(X_k) = Y_k
$$

(4.30)

and

$$
\nabla_{x\mu} \mathcal{L}(X_k, \mu_k) = -G^T_k
$$

where $G_k$ is the Jacobian of the constraints at $X_k$, i.e.

$$
G_k = \begin{bmatrix}
\nabla^T_x g_1(X_k) \\
\nabla^T_x g_2(X_k) \\
\vdots \\
\nabla^T_x g_m(X_k)
\end{bmatrix}
$$

The approximate KKT conditions in (4.25)-(4.28) can be expressed as

$$
Y_k \delta_x + f_k - G^T_k \mu_{k+1} = 0
$$

(4.31)

$$
G_k \delta_x \leq -g_k
$$

(4.32)

$$
\mu_{k+1} \geq 0
$$

(4.33)

$$
(\mu_{k+1})_i (G_k \delta_x + g_k)_i = 0 \quad \text{for} \ i = 1, \ldots, m
$$

(4.34)

where

$$
g_k = [g_1(X_k) \ g_2(X_k) \ \cdots \ g_m(X_k)]
$$

(4.35)
Equations (4.31)-(4.34) can be interpreted as the exact KKT conditions of the QP problem

\[
\text{Minimize} \quad \frac{1}{2} \delta^T Y_k \delta + \delta^T f_k \tag{4.36}
\]

\[
\text{Subject to} \quad G_k \delta \leq -g_k \tag{4.37}
\]

If \(\delta_x\) is a regular solution of the QP subproblem in (4.36)-(4.37) in the sense that the gradients of those constraints that are active at \(X_k\) are linearly independent, then equation (4.31) can be written as

\[
Y_k \delta_x + f_k - G^T_{ak} \hat{\mu}_{k+1} = 0
\]

where the rows of \(G_{ak}\) are those rows of \(G_k\) that are satisfying in the \((G_k \delta_x + g_k) \leq 0\) and \(\hat{\mu}_{k+1}\) denotes the associated Lagrange multiplier vector. Hence \(\hat{\mu}_{k+1}\) can be computed as

\[
\hat{\mu}_{k+1} = (G_{ak} G^T_{ak})^{-1} G_{ak} (Y_k \delta_x + f_k) \tag{4.38}
\]

It comes from the complementarity condition in (4.34) that the Lagrange multiplier \(\mu_{k+1}\) can be obtained by inserting zeros where necessary in \(\hat{\mu}_{k+1}\).

Since the key objective in the above method is to solve the QP subproblem in each iteration, the method is known as the **SQP method for general nonlinear minimization problems with inequality constraints** [4].

Algorithmically, the above SQP method is described as follows.

**Algorithm 4.2.**

- **Step 1** Initialize \(\{X, \mu\} = \{X_0, \mu_0\}\), \(X_0\) and \(\mu_0\) are chosen such that \(g_i(X_0) \leq 0\) for \(i = 1, \ldots, m\) and \(\mu_0 \geq 0\).

Set \(k = 0\) and initialize tolerance \(\epsilon\).
• **Step 2** Evaluate $Y_k$, $G_k$, $f_k$ and $g_k$ using (4.30) and (4.38).

• **Step 3** Solve the QP problem in (4.36)-(4.37) for $\delta_x$ and compute Lagrange multiplier $\mu_{k+1}$ using (4.38).

• **Step 4** Set $X_{k+1} = X_k + \delta_x$. If $\|\delta_x\| \leq \epsilon$, output $X^* = X_{k+1}$ and stop. Otherwise, set $k = k + 1$ and go to Step 2.

To apply above algorithm we consider following example.

**Example 4.2**

Minimize $f(X) = \frac{1}{2}[(x_1 - x_3)^2 + (x_2 - x_4)^2]$

Subject to:

$$\begin{align*}
- \frac{1}{4}x_1^2 - x_2^2 + \frac{1}{2}x_1 + \frac{3}{4} & \geq 0 \\
- \frac{5}{8}x_3^2 - \frac{5}{8}x_4^2 - \frac{6}{8}x_3x_4 + \frac{11}{2}x_3 + \frac{13}{2}x_4 - \frac{35}{2} & \geq 0
\end{align*}$$

with $X = [x_1 \ x_2 \ x_3 \ x_4]^T$ and $\mu_k = [\mu_1 \ \mu_2]^T > 0$, Step 2 of algorithm 4.2 gives:

$$G_k = \begin{bmatrix}
\nabla^T_x g_1(X_k) & \nabla^T_x g_2(X_k)
\end{bmatrix}^T$$

where $\nabla^T_x g_1(X_k) = \begin{bmatrix}
-1/2x_{1k} + 1/2 & 2x_{2k} & 0 & 0
\end{bmatrix}^T$

and $\nabla^T_x g_2(X_k) = \begin{bmatrix}
0 & 0 & -5/4x_{3k} - 3/4x_{4k} + 11/2 & -5/4x_{4k} - 3/4x_{3k} + 13/2
\end{bmatrix}^T$

$$Y_k = \begin{bmatrix}
1 + \frac{\mu_1}{2} & 0 & -1 & 0 \\
0 & 1 + 2\mu_1 & 0 & -1 \\
-1 & 0 & 1 + \frac{5\mu_2}{4} & \frac{3\mu_2}{4} \\
0 & -1 & 3\mu_4/4 & 1 + \frac{5\mu_2}{4}
\end{bmatrix}.$$ 

With $X_0 = [1.0 \ 0.5 \ 2.0 \ 3.0]^T$, $\mu_0 = [1 \ 1]^T$ and $\epsilon = 10^{-5}$, using the algorithm 4.2 after 7 iterations the algorithm converges to

$$X^* = \begin{bmatrix}
2.044750 \\
0.852716 \\
1.884720 \\
2.544913 \\
2.485633
\end{bmatrix}, \quad \mu^* = \begin{bmatrix}
0.957480 \\
1.100145
\end{bmatrix}.$$
and $f(X^*) = 1.45829$.

### 4.5 Stochastic Search Algorithm

In this section we explain the stochastic search algorithm. Stochastic search algorithm is not an iterative algorithm whereas, the methods which have mentioned in the previous sections are iterative some of them are exterior and else are interior point method. The stochastic search algorithm gives near-optimal solution to the problem. Near optimal solutions are useful when we satisfy with near-optimal solution which can be obtained quickly rather than exact optimal solution [27]. The proposed algorithm uses multiple starts to find the best near-optimal solution. If the convexity property of problem is there, then this near-optimal solution will be near to the global solution. The proposed algorithm is easy in programming and does not need to derivative mathematics knowledge.

We consider a nonlinear programming problem with inequality constraints where both objective function and constraints are nonlinear and smooth functions. It is assumed that the solution set is nonempty.

Consider following nonlinear programming problem.

\[
\begin{align*}
\text{Minimize} & \quad f(X) \quad X \in \mathbb{R}^n \\
\text{Subject to} & \quad g_i(X) \leq 0 \quad i = 1, \ldots, m, \\
& \quad X \geq 0.
\end{align*}
\]

As mentioned above, functions $f(X)$ and $g_i(X), i = 1, \ldots, m$ are continuous and twice differentiable functions. Also it is assumed that the feasible region $\omega$ that is described by constraints is nonempty.

We obtain the bounding box $B, B \subset E^n$ by considering the solution of the system of nonlinear equations. This system is the inequality nonlinear constraints
of the NLP problem which we consider it as a system of nonlinear equations with an initial value for solving this system. The elements of $B$ may be feasible solution to the problem or not. Then, generate a random element from $B$ using Uniform distribution. If this element is feasible, i.e. it is in $\Omega$, 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*.

The best solution at termination of the procedure is declared as the near-optimal solution.

### 4.5.1 Multiple starts

The result of stochastic search approach for every starting value is a near-optimal solution to the problem. Therefore, it may be useful 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 minimum/maximum of the objective function value obtained through the above procedure is **near-optimal** solution. 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 NLP problem:**
   
   Set $i = 1$, $z_{n-o} = 0$, $x_{n-o} = 0$, $i_0 = 100$ (number of feasible points),
   
   Number of restarting $= n_0$, Initial value for solving of nonlinear constraints
   
   equation system $= ix$.

2. **Generation in NLP problem:**
   
   **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. **Computation in NLP problem:**
   
   **Step 4** Compute $z_i = f(X_i)$ $i = 1, 2, \cdots, i_o$.
   
   **Step 5** If NLP is maximization, then $z_{n-o} \leftarrow \max_i(z_i), i = 1, 2, \cdots, i_o$.
   
   Otherwise, $z_{n-o} \leftarrow \min_i(z_i), i = 1, 2, \cdots, i_o$ and $x_{n-o} \leftarrow$ associated $x_i$
   
   with $z_{n-o}$.

4. **Termination in NLP problem:**
   
   **Step 6** Keep the values of $X_{n-o}$, $z_{n-o}$ and stop.

5. **Termination in restarting:** if the number of restarting get equal to its
   
   initial value, then output the values of $z_{n-o}$, $x_{n-o}$, $y_{n-o}$ and $y_{n-o}$ and stop restart.
4.5.2 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 \( \mu_j = \) upper bound of \( x_j \), \( j = 1, \ldots, n \) 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 by taken a fraction it. Another feature of the normal distribution is its unbounded support. This feature for the NLP problems that have bounded feasible region is not reasonable. For such this problems, 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 $0 < k < 1$.

4.6 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.

- $Z_{n-o}$ is the near-optimal objective functions.
- $\# \text{ inf. P.}$ is the number of infeasible points.
- $\# \text{ F.P. each It.}$ 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 for all iterations.
Example 4.3

\[ \text{Minimize} \quad f(X) = (1 - x_1)^2 - 10(x_2 - x_1^2)^2 + x_1^2 - 2x_1x_2 + e^{-x_1-x_2} \]

\[ \text{Subject to} \quad x_1^2 + x_2^2 \leq 16 \]
\[ (x_2 - x_1)^2 + x_1 \leq 6 \]
\[ x_1 + x_2 \leq 2 \]

The optimal objective function value is -154.8647. The summary of results are shown in the Tables 4.1-4.3. We have considered the \(ix = [1 \quad 1]^T\) as initial solution for solving of nonlinear constraints equation of NLP problem.

**Table 4.1** Near-Optimal values of objective function in above NLP problem by implementing proposed stochastic search algorithm with considering Uniform and Normal distributions.

<table>
<thead>
<tr>
<th>Distribution</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>(Uniform(0, \mu_j))</td>
<td>-147.96</td>
<td>247</td>
<td>2278</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>(N(0, 1))</td>
<td>-139.725</td>
<td>386</td>
<td>4425</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>(N(0, \mu_j^2))</td>
<td>-147.194</td>
<td>2395</td>
<td>2030</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>(N(\mu_j, \mu_j^2))</td>
<td>-144.967</td>
<td>3333</td>
<td>32564</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>(N(1/2\mu_j, \mu_j^2))</td>
<td>-148.122</td>
<td>3402</td>
<td>33126</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 4.2 Near-Optimal values of objective function in above NLP problem by implementing proposed stochastic search algorithm with considering Beta and Gamma distributions.

<table>
<thead>
<tr>
<th>Distribution</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></td>
<td></td>
<td>IPBS</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mu_j \ast Beta(0.5, 0.5)$</td>
<td>$-148.333$</td>
<td>184</td>
<td>1834</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Beta(0.75, 0.5)$</td>
<td>$-151.667$</td>
<td>305</td>
<td>3621</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Beta(0.5, 0.75)$</td>
<td>$-153.742$</td>
<td>103</td>
<td>992</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Beta(0.5, 0.5)$</td>
<td>$-148.333$</td>
<td>184</td>
<td>1834</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Beta(0.05, 0.2)$</td>
<td>$-154.383$</td>
<td>27</td>
<td>251</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Gamma(0.5, 1)$</td>
<td>$-150.426$</td>
<td>108</td>
<td>855</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Gamma(1, 1)$</td>
<td>$-144.404$</td>
<td>506</td>
<td>21610</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Gamma(2, 1)$</td>
<td>$-143.24$</td>
<td>1118</td>
<td>111853</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j \ast Gamma(1, 2)$</td>
<td>$-147.19$</td>
<td>1655</td>
<td>15509</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.3 Near-Optimal values of objective function in above NLP problem by implementing proposed stochastic search algorithm with considering Geometric and Poisson distributions.

<table>
<thead>
<tr>
<th>Distribution</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></td>
<td></td>
<td>IPBS</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geometric(0.5)</td>
<td>$-154.864$</td>
<td>51</td>
<td>449</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>Poisson(1)</td>
<td>$-154.864$</td>
<td>61</td>
<td>463</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

- As can be seen from Table 4.1, the Uniform and Normal distribution don’t have reasonable efficiency. But in the normal distribution when we consider a fraction of mean as mean parameter the objective value get nearer to the optimal objective value.

- From Table 4.2 can be understood that Beta distribution is more efficient than Gamma distribution. In this example the feasible region is bounded.

- The discrete distributions (Geometric and Poisson) produce the exact opti-
mal solution because the optimal solution is integer.

Example 4.4

Maximize \( x_1^3 + 2x_2^2x_3 + 2x_3 \)

Subject to \( x_1^2 + x_2 + x_3^2 \leq 4, \)
\( x_1^2 - x_2 + 2x_3 \leq 2, \)
\( x_1 \geq 0, \ x_2 \geq 0, \ x_3 \geq 0. \)

The optimal objective function value is 20.14168. The summary of results are shown in the Tables 4.4 and 4.5. We have considered the \( ix = [0 \ \ 3 \ \ 1]^T \) as initial solution for solving of nonlinear constraints equation of NLP problem.

Table 4.4 Near-Optimal values of objective function in above NLP problem by implementing proposed stochastic search algorithm with considering Uniform, Normal and Weibull distributions.

<table>
<thead>
<tr>
<th>Distribution</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>Uniform(0, ( \mu_j ))</td>
<td>8.3945</td>
<td>20</td>
<td>217</td>
<td>100</td>
<td>0.45</td>
</tr>
<tr>
<td>N(0,1)</td>
<td>16.4</td>
<td>1159</td>
<td>12479</td>
<td>100</td>
<td>10.061</td>
</tr>
<tr>
<td>N(0, ( \mu_j^2 ))</td>
<td>19.283</td>
<td>613</td>
<td>6191</td>
<td>100</td>
<td>0.5</td>
</tr>
<tr>
<td>N(( \mu_j,\mu_j^2 ))</td>
<td>20.1315</td>
<td>487</td>
<td>4630</td>
<td>100</td>
<td>0.615</td>
</tr>
<tr>
<td>N(1/2( \mu_j,\mu_j^2 ))</td>
<td>20.04</td>
<td>273</td>
<td>2544</td>
<td>100</td>
<td>0.561</td>
</tr>
<tr>
<td>Weibull(( \mu_j,1 ))</td>
<td>19.887</td>
<td>96</td>
<td>960</td>
<td>100</td>
<td>0.31</td>
</tr>
<tr>
<td>Weibull(2( \mu_j,2 ))</td>
<td>20.026</td>
<td>709</td>
<td>6752</td>
<td>100</td>
<td>1.225</td>
</tr>
</tbody>
</table>

Table 4.5 Near-Optimal values of objective function in above NLP problem by implementing proposed stochastic search algorithm with considering Gamma distribution.
Distribution | $Z_{n-o}$ | #inf. P. | # F. P. | # Re starts | CPU time
|---|---|---|---|---|---
| $\mu_j * \text{Gamma}(1,1)$ | 19.971 | 78 | 933 | 100 | 10 | 0.6347
| $\mu_j * \text{Gamma}(2,1)$ | 19.845 | 465 | 5747 | 100 | 10 | 1.112
| $\mu_j * \text{Gamma}(2,3)$ | 20.1247 | 18157 | 150175 | 100 | 10 | 13.6

- From Table 4.4 the performance of Uniform distribution is not good but Normal distribution with mean=\(\mu_j\) and variance=\(\mu_j^2\) is good. The kurtosis of this normal distribution is lower than others which are mentioned in the table. Also the performance of Weibull distribution is relatively like Normal distribution.

- From Table 4.5 Gamma distribution for this NLP problem is suitable. The best distribution for this NLP problem is $N(\mu_j, \mu_j^2)$.

**Example 4.5**

$$\text{Minimize} \quad f(X) = (x_1 - 2)^2 + (x_2 - 1)^2$$

**Subject to**

$$-x_1^2 + x_2 \geq 0,$$

$$-x_1 + x_2^2 \geq 0.$$  

The local optimal objective function value is 1.00. The summary of results are shown in the Table 4.6 and 4.7. We have considered the $ix = [2 \ 2]^T$ as initial solution for solving of nonlinear constraints equation of NLP problem.
Table 4.6 Near-Optimal values of objective function in above NLP problem by implementing proposed stochastic search algorithm with considering Uniform and Normal distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{n-o}$</th>
<th>#inf. P. IPBS</th>
<th># F. P. each It.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform(0, $\mu_j$)</td>
<td>1.02</td>
<td>192</td>
<td>1800</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$N(0, 1)$</td>
<td>0.8718</td>
<td>984</td>
<td>9551</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$N(0, \mu^2_j)$</td>
<td>0.8795</td>
<td>909</td>
<td>9979</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$N(\mu_j, \mu^2_j)$</td>
<td>0.827</td>
<td>245</td>
<td>2707</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$N(1/2\mu_j, \mu^2_j)$</td>
<td>0.8361</td>
<td>396</td>
<td>3881</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.7 Near-Optimal values of objective function in above NLP problem by implementing proposed stochastic search algorithm with considering Beta and Gamma distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$Z_{n-o}$</th>
<th>#inf. P. IPBS</th>
<th># F. P. each It.</th>
<th># Re starts</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_j * Beta(0.5, 0.5)$</td>
<td>1.0025</td>
<td>173</td>
<td>1531</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.5, 0.75)$</td>
<td>1.0084</td>
<td>204</td>
<td>1841</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.75, 0.5)$</td>
<td>1.0008</td>
<td>135</td>
<td>1535</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Beta(0.05, 0.05)$</td>
<td>1.00</td>
<td>113</td>
<td>1174</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(1, 1)$</td>
<td>0.8359</td>
<td>172</td>
<td>1460</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(0.5, 1)$</td>
<td>0.8640</td>
<td>161</td>
<td>1783</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(1, 2)$</td>
<td>0.8478</td>
<td>174</td>
<td>1553</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(2, 1)$</td>
<td>0.8344</td>
<td>242</td>
<td>1920</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\mu_j * Gamma(2, 2)$</td>
<td>0.8426</td>
<td>313</td>
<td>3148</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

• The deterministic algorithms like interior-point algorithm for nonconvex problems produce a local optimal, (minimum), objective value equal to 1.00 [4].

Whereas, non-uniform distributions like Normal and Gamma give an near-optimal objective value which is less than 1.00.

• Previous item aims that there is no risk to fall in a local optimum.
• Normal and Gamma distributions have good efficiency. The feasible region is nonconvex, the uniform distribution and supported distributions don’t have less efficiency than others.

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