GANIT *J. Bangladesh Math. Soc.* (ISSN 1606-3694) **35** (2015) 41-55

AN ALGORITHMIC TECHNIQUE FOR SOLVING NON-LINEAR PROGRAMMING AND QUADRATIC PROGRAMMING PROBLEMS

H. K. Das[1] and M. Babul Hasan
Department of Mathematics, University of Dhaka, Dhaka-1000, Bangladesh
[1]Corresponding author: hkdas_math@du.ac.bd

Received 05.09.2014 Accepted 28.03.2015

ABSTRACT

In this paper, we improve a combined algorithm and develop a uniform computer technique for solving constrained, unconstrained Non Linear Programming (NLP) and Quadratic Programming (QP) problems into a single framework. For this, we first review the basic algorithms of convex and concave QP as well as general NLP problems. We also focus on the development of the graphical representations. We use MATHEMATICA 9.0 to develop this algorithmic technique. We present a number of numerical examples to demonstrate our method.

Keywords: Constrained, Unconstrained, Optimization, Non Linear Programming, Quadratic Programming, Computer Algebra

1. Introduction

In the literature of operation research a linearly or nonlinearly constrained and unconstrained problem with a linear or nonlinear objective functions has many applications; this is often viewed as a discipline in and of itself. NLP is a mathematical technique for determining the optimal solutions to many business problems. Because of its usefulness in production planning, financial and corporate planning, health care and hospital planning of QP & NLP problems have attracted considerable research and interest in recent years. It has become an important branch of operation research and has a wide variety of applications in such areas as the military, economics, engineering optimization and management sciences (Nash and Sofer 1996). There is an extremely diverse range of practical applications. Ayoade[1]said in his paper that “It is impossible to develop a single algorithm that will solve all optimization problems efficiently and as such a number of methods have been developed”. Unlike the simplex method and basis matrix for LP, no single algorithm is available to solve all these different types of problems. So, our focus is to develop a uniform algorithm and its uniform computer technique that will be able to solve different type of NLP and QP problems into a single framework. However, recently there are several researchers who worked on the computer development of different types of NLP problems. In 2012, Datta & Hasan[6,7] developed computer technique for the Lagrange method and Karush-Kuhn-Tucker Method. Later, in 2013, Das and Hasan [4] developed a generalized computer technique for solving unconstrained NLP problems. Recently, in 2014, Das, Saha & Hasan [5] have studied on the 1-D Simplex Search and showed some merits and demerits in 1-D methods through numerical experiments. But all of them are silent about QP problems and the uniform computer technique. In this paper, we combine those computer techniques with some modifications and then developed a uniform computer technique for solving NLP and QP problems. Now-a-days, the world is being ruled by the fastest. So, we must try to finish our job as fast as we can. Moreover, hand calculation is very tough and time consuming for analyzing the NLP and QP problems with large number of variables and constrains. In this paper, our aim is to develop a uniform method whereas; one does not need to be confused about the type of the NLP problems.

We now consider the general optimization problem to minimize or maximize the objective function under unconstraint, nonlinear equality or inequality type constraints. The objective of this research is to evaluate the performance of some existing algorithms for solving NLP and QP problems. For reviewing the existing algorithm we need to develop its computational technique. It is extremely difficult to state that one method is definitely superior to another method without analytical or numerical experimental investigation. On the other hand, in this research, we find the prescribed paper [4, 5, 6, 7] which is successful in particular type of NLP problems and so we have modified this algorithm into a single framework successfully by developing a new computer technique. Finally, we develop a computer technique for those types of NLP problems. It is important that all the test examples come with an optimal solution obtained by analytical or numerical experimental investigation. For this reason, we have evaluated the performance of all the algorithms by the computational way in attempting to solve a set of test problems.

The rest of the paper is organized as follows. In Section 2, we briefly discuss some existing methods with its merits and demerits on consideration to this paper. In Section 3, we develop an algorithmic technique and its computer coding for solving NLP and QP problems using *MATHEMATICA* [8, 13]. We have shown the steps of the algorithm and program in a table so that it can be visualize instantly what method is applied there. In Section 4, we present the comparisons, results and discussion among the selected test problems (TP).

2. Existing Methods

In this section, we focus on the some existing methods for solving NLP and QP with its merits and demerits in the real world problems. We then discuss precisely the formulation of different type of NLP problems. We also briefly discuss the formulation of QP and NLP problems which has shown the necessity for developing a uniform technique.

2.1 One Dimensional Method of NLP [H. K. Das 4, 5]

In this section, we discuss the 1-D simplex search methods for finding the optimum solutions and also present two flowcharts of Choo & Kim two phase method in H. K. Das & Hasan [4].

2.1.1 Choo & Kim 1-D Simplex Search

In 1987 this method was proposed by Choo and Kim which is one of the most robust and efficient 1-D direct search methods which had been analyzed in H. K. Das and Hasan [3, 4].

i. This method can start with any two points and converges to the optimum solution.

ii. With appropriate parameters, this algorithm can be made to behave equivalent to some of the most efficient 1-D search methods.

2.1.2 Powels 1-D Simplex Search

In here, we summarize the "Powels" 1-D simplex search method which is given as follows:

i. Particular attention was given to the contribution of theoretical analysis.

ii. Reviews some of the most successful methods for unconstrained, constrained and non differentiable optimization calculations.

2.1.3 Golden Section & Fibonacci Search

In this section, we summarize the method of "Golden Section" search method. In 1953 "Golden section" and "Fibonacci search" was developed by Kiefer which is given as follows:

i. Find the extremum of a unimodal function over an interval without using derivatives.

ii. Golden section narrows the range of values and it is based on the golden ratio.

iii. If the interval is not optimal then the method will be failand needs much iteration.

2.1.3 Nelder and Mead (NM) Method

In 1965 Nelder and Mead methods were proposed by Nelder and Mead which discuss as follows:

i. It designed for unconstrained optimization without using gradients.

ii. Operation of this method is to rescale on the local behavior of the function by using four basic procedures: 1. Reflection 2. Expansion 3. Contraction 4. Shrinking

2.2 Karush-Kuhn-Tucker Method [6]

Karush-Kuhn-Tucker (KKT) conditions for the nonlinear program with its merits and demerits. Let z be a real valued function of n variables defined by $z = f(x_1, x_2, \dots\dots\dots\dots x_n)$ and $\{b_1, b_2, \dots\dots\dots\dots b_m\}$ be a set of right hand side constants of (1). If either $f(x_1, x_2, \dots\dots\dots\dots x_n)$ or some $g^i(x_1, x_2, \dots\dots\dots\dots x_n)$, $i = 1,2, \dots\dots\dots m$; or both are non-linear, then the problem of determining the n-type $(x_1, x_2, \dots\dots\dots\dots x_n)$ which makes z a maximum or minimum and satisfies the following conditions, is called a general NLP problem such that

$$\begin{cases} g^1(x_1, x_2, \ldots, x_n) \{\le, \ge \text{ or } =\} b_1 \\ g^2(x_1, x_2, \ldots, x_n) \{\le, \ge \text{ or } =\} b_2 \\ \ldots \quad \ldots \quad \ldots \quad \ldots \\ \ldots \quad \ldots \quad \ldots \quad \ldots \\ g^m(x_1, x_2, \ldots, x_n) \{\le, \ge \text{ or } =\} b_m \end{cases} \qquad (1)$$

where g^i's are real valued functions of n variables $x_1, x_2, \ldots, x_n$ and $x_j \ge 0, j = 1,2, \ldots m$;

1. This method can be used to solve NLP's in when all the constraints are not equal.
2. This method fails when the constraints are equal.

In the following , a theorem (a table) is given to visualize the standard form of KKT that we have used in our algorithm.

Theorem 1:

Assume that $f(x), g_1(x), g_2(x), \ldots \ldots \ldots \ldots g_m(x)$ are differentiable functions satisfying certain regularity condition. Then $x^* = (g_1{}^*, g_2{}^*, \ldots \ldots \ldots g^*{}_n)$ can be an optimal solution for the NLP only if there exist m equations such that all the following KKT conditions are satisfied. (Wayne L. Winston [14]

Table 1: Standard Form of KKT

1.	$\frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} \le 0$	
2.	$x^*{}_j \left(\frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} \right) = 0$	
3.	$g_i(x^*) - b_i \le 0$	for $i = 1,2, \ldots \ldots \ldots . m.$
4.	$u_i[g_i(x^*) - b_i] = 0$	for $i = 1,2, \ldots \ldots \ldots . m.$
5.	$x^*{}_j \ge 0,$	for $j = 1,2, \ldots \ldots \ldots . m.$
6.	$u_i \ge 0,$	for $i = 1,2, \ldots \ldots \ldots . m.$

2.3 Lagrange's Method [7]

The Lagrange's has been discussed on the following way:

1. Lagrange multipliers can be used to solve NLP's in which all the constraints are equal.
2. To solve equal constraints we associate a multiplier λ_i with the i-th constraint.

In general construction, we focus on way of recognition an optimal solution for NLP problem with equality constraints. Lagrange multipliers can be used to solve such problems. We consider NLP's of the following types:

$$\left.\begin{aligned}
&\text{Max (or Min): } z = f(x_1, x_2, \dots\dots\dots\dots . x_n)\\
&\quad \text{s/t, } g_1(x_1, x_2, \dots\dots\dots\dots . x_n) = b_1\\
&\qquad g_2(x_1, x_2, \dots\dots\dots\dots . x_n) = b_2\\
&\qquad g_m(x_1, x_2, \dots\dots\dots\dots . x_n) = b_m
\end{aligned}\right\} \qquad (1')$$

To solve $(1')$, we associate a multiplier λ_i with the i-th constraint in $(1')$ and hence Lagrangian

$L(x_1, x_2, \dots\dots\dots\dots . x_n, \lambda_1, \lambda_2, \dots\dots\dots\dots \lambda_m) = f(x_1, x_2, \dots\dots\dots\dots . x_n) + \sum_{i=1}^{i=m} \lambda_i [b_i - g_i(x_1, x_2, \dots\dots\dots\dots . x_n)] \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$

Then we attempt to find a point $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots\dots\dots .. \bar{\lambda}_m)$ that maximizes (or minimize) $L(x_1, x_2, \dots\dots\dots\dots . x_n, \lambda_1, \lambda_2, \dots\dots\dots\dots . \lambda_m)$. In many situations $(\bar{x}_1, \bar{x}_2, \dots \bar{x}_n)$

will solve $(1')$. Suppose that $(1')$ is a maximization problem. If $(x_1, x_2, \dots\dots x_n, \lambda_1, \lambda_2, \dots\dots \lambda_n)$

maximizes L then at $(x_1, x_2, \dots\dots\dots\dots . x_n, \lambda_1, \lambda_2, \dots\dots\dots\dots . \lambda_n)$

$$\frac{\partial L}{\partial \lambda_i} = b_i - g_i(x_1, x_2, \dots\dots\dots\dots . x_n) = 0$$

Here $\frac{\partial L}{\partial \lambda_i}$ is the partial derivative of L with respect to λ_i. This shows that $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n)$ will satisfy the constraints in $(1')$.

To show that $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n)$ solves $(1')$, let $\acute{x}_1, \acute{x}_2, \dots\dots\dots\dots . \acute{x}_n$ be any point that is in $(1')$'s feasible region. Since$(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots\dots\dots .. \bar{\lambda}_m)$ maximizes L, for any numbers $\acute{\lambda}_1, \acute{\lambda}_2, \dots\dots\dots .. \acute{\lambda}_m$

we have,

$$L(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots\dots\dots .. \bar{\lambda}_m) \geq L(\acute{x}_1, \acute{x}_2, \dots\dots\dots\dots . \acute{x}_n, \acute{\lambda}_1, \acute{\lambda}_2, \dots\dots\dots .. \acute{\lambda}_m) \dots\dots (3)$$

Since $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n)$ and $(\acute{x}_1, \acute{x}_2, \dots\dots\dots\dots . \acute{x}_n)$ are both feasible in$(1')$, the terms in

we have,

$$L(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots\dots\dots .. \bar{\lambda}_m) \geq L(\acute{x}_1, \acute{x}_2, \dots\dots\dots\dots . \acute{x}_n, \acute{\lambda}_1, \acute{\lambda}_2, \dots\dots\dots .. \acute{\lambda}_m) \dots\dots (3)$$

Since $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n)$ and $(\acute{x}_1, \acute{x}_2, \dots\dots\dots\dots . \acute{x}_n)$ are both feasible in$(1')$, the terms in

(3) involving the $\lambda's$ are all zero, and now (3) becomes

$f(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n) \geq f(\acute{x}_1, \acute{x}_2, \dots\dots\dots\dots . \acute{x}_n)$.

Thus, $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n)$ does solve$(1')$. In short, if $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots\dots\dots .. \bar{\lambda}_m)$ shows the unconstrained maximization problem Maximize $L(x_1, x_2, \dots x_n, \lambda_1, \lambda_2, \dots \lambda_m) \dots\dots\dots\dots\dots\dots (4)$

Then $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n)$ solves $(1')$.We know that for $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots\dots\dots .. \bar{\lambda}_m)$ to solve (4) is necessary that at $(\bar{x}_1, \bar{x}_2, \dots\dots\dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots\dots\dots .. \bar{\lambda}_m)$,

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial x_2} = \cdots \frac{\partial L}{\partial x_n} = \frac{\partial L}{\partial \lambda_1} = \frac{\partial L}{\partial \lambda_2} \dots\dots\dots\dots = \frac{\partial L}{\partial \lambda_m} = 0 \dots\dots\dots\dots\dots\dots\dots\dots (5).$$

Theorem 2:

Suppose $(1')$ is a maximization problem. If $f(x_1, x_2, \dots \dots \dots x_n)$ is a concave function and each $g_i(x_1, x_2, \dots \dots \dots x_n)$ is a linear function, then any point $(\bar{x}_1, \bar{x}_2, \dots \dots \dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots \dots \dots .. \bar{\lambda}_m)$ satisfying (5) will yield an optimal solution $(\bar{x}_1, \bar{x}_2, \dots \dots \dots \bar{x}_n)$ to (1)

The above theorem $(1')$ gives conditions implying that any point $(\bar{x}_1, \bar{x}_2, \dots \dots \dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots \dots \dots .. \bar{\lambda}_m)$ that satisfying (5) will yield an optimal solution $(\bar{x}_1, \bar{x}_2, \dots \dots \dots \bar{x}_n)$ to $(1')$, Wayne L. Winston [14].

Theorem 3:

Suppose $(1')$ is a minimization problem. If $f(x_1, x_2, \dots \dots \dots x_n)$ is a convax function and each $g_i(x_1, x_2, \dots \dots \dots x_n)$ is a linear function, then any point $(\bar{x}_1, \bar{x}_2, \dots \dots \dots \bar{x}_n, \bar{\lambda}_1, \bar{\lambda}_2, \dots \dots \dots .. \bar{\lambda}_m)$ satisfying (5) will yield an optimal solution $(\bar{x}_1, \bar{x}_2, \dots \dots \dots \bar{x}_n)$ to$(1')$.Even if the hypothesis of these theorems fail to hold, it is possible that any point satisfying (5) will solve$(1')$ Wayne L. Winston [14].

2.4 Swarup's Simplex Type Method for Solving QP [10, 12]

The iterative procedure for the solution of a QP problem by Swarup's Simplex Type Method can be summarized as follows:

1. It is applicable only special type of QP.
2. Adding slack variables $X_{n+i},\ i = 1,2, \dots .. m$ to the constraints of (1), we have (QPI): Maximize $z = (cx + \alpha)(dx + \beta)$

   Subject to: $Ax = b, x \geq 0$ where $A, b, c, d, x, \alpha, \beta$ are defined.
3. Optimality Condition :The value of the objective function we will improve If $\hat{Z} > Z$ $\therefore Z^2(C_u - {Z_u}^1) + Z^1(d_u - {Z_u}^2) + \theta_u(C_u - {Z_u}^1)(d_u - {Z_u}^2) > 0$ where $\theta_u = \hat{X}_{B_r}$
4. The solution can be improved until $R_j \leq 0$ for all $j = 1,2, \dots .. n$,when ever all $R_j \leq 0$ for all $j = 1,2, .. n$ at a simplex table,

   Where,$R_j(say) = Z^2(C_j - {Z_j}^1) + Z^1(d_j - {Z_j}^2) + \theta_j(C_j - {Z_j}^1)(d_j - {Z_j}^2)$

The solution becomes optimal and the process terminates.

Criterion1: (Choice of the entering variable).

Criterion 2: (Choice of the outgoing variable).

To develop a uniform method by implement the existing methods, we have modified or extended the existing methods for solving all type of NLP and QP problems in a single framework. In Table 2, we are going to present the Necessary and Sufficient conditions of the optimality for the NLP problems.

Table 2. Necessary and Sufficient conditions for optimality

Problem Type	Necessary Conditions for Optimality	Also Sufficient If:
One-variable unconstrained	$\frac{dy}{dx} = 0$	$f(x)$ concave
Multivariable unconstrained	$\frac{\partial f}{\partial x_j} = 0 \ (j = 1,2,\dots,n)$	$f(x)$ concave
Constrained, nonnegative constraints only	$\frac{\partial f}{\partial x_j} = 0 \ (j = 1,2,\dots,n)$ (Or ≤ 0 if $x_j = 0$)	$f(x)$ concave
General Constrained Problem	Karush-Khun-Tucker conditions	$f(x)$ concave and ${}^{g}_{i}(x)$ convex$(i = 1,2,\dots,m)$

3. Algorithmic Technique

In here, we first develop a uniform algorithm for solving NLP and QP problems and we then develop a code using programming language MATHEMATICA [7,12].

3.1 Uniform Algorithmic Technique For Solving NLP and QP

In this section, we combine relevant existing algorithms for solving NLP and QP problems. In our method firstly we separate the type of the NLP problems and then combine the different type of existing NLP problems. The whole algorithm steps proceed as follows:

Step1: Input number of variables (v), number of inequality constraints (vv), number of equality constraints (vvv).

Step 2: If number of variables v=1, number of inequality constraints vv=0, number of equality constraints vvv=0 then go to the following Step.

Step 3: Finding the optimum interval using the following Sub Steps.

Sub-Step 1: Select ant two points b_1 and w_1 such that $f(w_1) \geq f(b_1)$. Set $k = 1$.

Sub-Step 2: Contraction: $c = b_1 + \beta(w_1 - b_1)$. If $f(c) \leq f(b_1)$ then set $m = b_1$, $h = c$, $s = w_1$ and stop. Otherwise go to Sub-step 3.

Sub-Step 3: Reflection: $r_k = b_k + \alpha(b_k - w_k)$.

Sub-Step 4: If $f(r_k) \geq f(b_k)$ then go to Sub-Step 5.

Expansion: $e_k = b_k + (\alpha + \delta)(w_k + b_k)$. If $f(e_k) > f(r_k)$ then set $w_{k+1} = b_k$, $b_{k+1} = r_k$, $r_{k+1} = e_k$, $k = k + 1$ and go to Sub-Step 5. Otherwise set $w_{k+1} = b_k$, $b_{k+1} = r_k$, $k = k + 1$ and go to Sub-Step 3.

Sub-Step 5: Set $h = b_k$ if $f(w_k) \leq f(r_k)$ then set $m = w_k$, $s = r_k$ and stop and go to step-4. Otherwise set $m = r_k$, $s = w_k$ and go to the step 3.

Step 4: *To find the* optimal solution $\bar{x}$ of the NLP we use the value $m = r_k$, $s = w_k$ and follow the following Sub-Steps.

Sub-Step 1: Set k=0 and go to *Sub*-Step 2.

Sub-Step 2: Reflection: let $r_k = h_k + \alpha(h_k - m_k)$. go to *Sub*-Step 3.

Sub-Step 3: If $f(r_k) \geq f(h_k)$ then go to *Sub*-Step 5.

Expansion: $e_k = h_k + (\alpha + \delta)(h_k + m_k)$. If $f(e_k) > f(r_k)$ then go to sub-step 4. Otherwise let $h_{k+1} = e_k$, $m_{k+1} = r_k$, $s_{k+1} = s_k$, $k = k + 1$ and go to Sub-Step 2.

***Sub-Step* 4:** If $f(r_k) \geq f(h_k)$ then let $h_{k+1} = e_k$, $m_{k+1} = r_k$, $s_{k+1} = s_k$, $k = k + 1$ and go to *Sub*-Step 2. Otherwise, let $h_{k+1} = r_k$, $m_{k+1} = e_k$, $s_{k+1} = h_k$, $k = k + 1$ and go to *Sub*-Step 2.

***Sub-Step* 5:** Contraction: if $f(r_k) \geq f(m_k)$ then let $h_{k+1} = h_k$, $m_{k+1} = r_k$, $s_{k+1} = m_k$, $k = k + 1$ and go to *Sub*-Step 2. Otherwise, we let $c_k = h_k + \beta(m_k - h_k)$. If $f(c_k) \leq f(h_k)$ then let $h_{k+1} = h_k$, $m_{k+1} = c_k$, $s_{k+1} = r_k$, $k = k + 1$ and go to *Sub*-Step 2. Otherwise, we let $h_{k+1} = c_k$, $m_{k+1} = h_k$, $s_{k+1} = m_k$, $k = k + 1$ and go to *Sub*-Step 2.

Sub-Step (6): Stop and optimal solution find.

Step 5: If number of variables v≥2, number of inequality constraints vv=0, number of equality constraints vvv=0 then go to the following Sub-Step.

Sub-Step 1: Select ε and any initial trial solution x'. Max $Z = f(x), x \in \mathbb{R}^+ \cup \{0\}$.

Sub-Step 2: Express $f(x' + \nabla f(x'))$ as a function t by setting$x_j = x_j' + t(\frac{\partial f}{\partial x_j})_{x=x'}$ and

Sub-Step 3: Using the one dimensional search procedure find $t = t^*$ s.t.

$f\left(x' + t\nabla f(x')\right)$is maximized over $t \geq 0$

Sub-Step 4: Calculate$\nabla f(x')$ at new x' If $\nabla f(x') \leq \varepsilon$ i.e. If $\frac{\partial f}{\partial x_i} < \varepsilon$ then stop.

Step 6: If number of variables $v \geq 1$, number of inequality constraints vv≠0, number of equality constraints vvv=0 then go to the following Sub-Step.

Sub-Step 1: Input number of constraints (n), number of variables (m) and the unknowns as $\{x_1, x_2, \ldots \ldots \ldots . x_n\}$, objective function (f) and the constraints $(g_i, i = \overline{1,2, \ldots . . n})$ in term of unknowns.

Sub-Step 2: Input mm, for maximization input *0* *&*for minimization input *1*.

Sub-Step 3: Define "Lagrange". If $mm = 0$ set $l = f - \sum_{i=1}^{i=n} u_i g_i$ else set $= f + \sum_{i=1}^{i=n} u_i g_i$.

Sub-Step 4: *S*et eqs of $u_i g_i (i = \overline{1,2, \ldots \ldots n})$, $\frac{\delta l}{\delta x_i} (i = \overline{1,2, \ldots \ldots n})$

Sub-Step 5: Sol = Solve [eqs].

Sub-Step 6: Discard the solutions from sol for which $g_i > 0$ or $u_i < 0$.

Sub-Step 7: Print feasible solution sol.

Sub-Step 8: Calculate objective function value for each elements of sol.

Sub-Step 9: For *mm* = 0 find maximum value of objective functions and their corresponding index or *mm* = 1 find minimization value of objective functioins and their corresponding index.

Sub-Step 10: Print solution corresponding to index and the objective functional value.If not, go to step 7.

Step 7: If number of variables $v \geq 1$, number of inequality constraints vv≠0, number of equality constraints *vvv* ≠ 0 then go to the following Sub-Step.

Sub-Step 1: Input number of variables (m), number of constraints (n), the unknowns

*a*s $\{x_1, x_2, x_3, \ldots, x_n\}$, objective function (f).

Sub-step 2: For maximization input zero (0) and for minimization input one (1).

Sub-Step 3: If o = 0; set $l = f - \sum_{i=1}^{i=n} u_i g_i$ else set $l = f + \sum_{i=1}^{i=n} u_i g_i$ *Sub-Step 4 :* Solve $\frac{\partial f}{\partial x_i} =$ 0 for *j* = 1, … m and solve $\frac{\partial f}{\partial x_j} = 0$, for *i* = 1, …, *n*.

Sub-Step 5: Set sol in functional (f).

Sub-Step 6: Print solution and objection functional value.

Stop.

3.2 Program Organization

In this section, we introduce a uniform computer technique and we will use five module functions **MODIFIED[PHASE0_]**, **PHASE[1_]**, **MA[GRADIENT_], VOGOB[KT_] and MAA[Lag_].** The main module function is **main [unconcons_]** which call all the module functions.

3.2.1 Unique Computer Techniques

For any new technique, sometimes it comes urgency to make programming code to verify lots of different test problems whether the method is subject to generalized or not. Considering this notion in care, we just want to view the programs coded by us using the programming language *MATHEMATICA* corresponding to our unique algorithm.

In the current section, we develop a generalized code for solving NLP problems using MATHEMATICA [7] corresponding to the algorithm in Section 3.1. Our develop computer technique is not presented here for the page limitation. But, if the readers are interest to observe the reliability of our developed code then please contact with authors via editor.

3.3 Programming Input and Output Organization

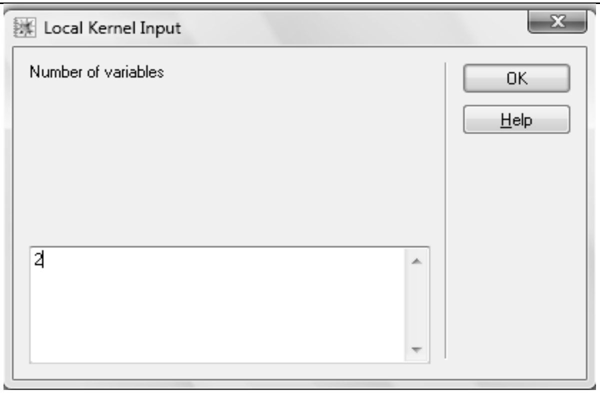
In the current section, we will present the programming input and output information and how to build up the graphical representations of the NLP problems.

3.3.1 Programming Input

Our input information corresponding to the program in Section 3.2.1 to evaluate the performance of the unique algorithm in Section 3.1 and our developed code is visualized in this section. The following table is the input of the module function. If we run the coded program by pressing the "Enter" from the Key board then we will get the following "Local kernel Input" box.

On the screen of Local kernel Input (LKI), it shows that number of variables, number of inequality constraints, number of equality constraints. We consider the example 5 where variables are in the form $\{x_1, x_2\}$ and constraints are given in the form $4x_1 + 3x_2 - 16$.

Table 3. Coding Input Information.

Module Functions	Fig 1: Local Kernel Input Box
MODIFIED[PHASE0_] **PHASE[1_]** **MA[GRADIENT_]** **VOGOB[KT_]** **MAA[Lag_]** **main [unconcons_]**	 2

3.3.2 Programming Output

In this paper, we not only summarize the existing methods but also develop a uniform computer technique which can solve all type of NLP and QP problems. The Input and output information of a number of test problems is given below. We also show the workability of our technique. To observe the differences between our technique and the existing techniques, we solve the same problem which was given in Table 3 and obtain the same solutions. The main difference is that those methods can solve only particular type of problems but our technique works for all NLP and QP problems.

For considering brevity throughout this paper, we have chosen some selected test problem (TP) in the following from the problem list in Section 4.1.

Test Problem Number 1

Input

main[unconcons]

Output

Final solution is given below:

{0.999992, 0.999992} 1. 34

$\left\{\frac{1}{65536}, 0\right\}$ $\left\{\frac{65535+t}{65536}, \frac{131071}{131072}\right\}$ $\left\{\left\{t \to \frac{1}{2}\right\}\right\}$

{0.999992, 0.999996} 1. 35 $\left\{0, \frac{1}{65536}\right\}$

$\left\{\frac{131071}{131072}, \frac{131071+2t}{131072}\right\}$ $\left\{\left\{t \to \frac{1}{4}\right\}\right\}$

After 35 iterations it gives the solution approximate to (1,1) and the optimal value approximate to 1.

Test Problem Number 2

Input

main[unconcons]

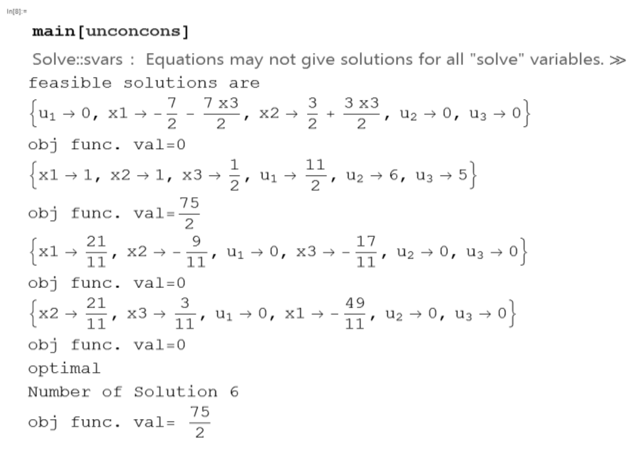
Output:

No. of Iteration in **modified phase 0** is 1 and No. of Iteration in **phase 1** 21. Total iteration is 22 whaereas if we use the Cho & Kim Phase 0 and Phase1 then the total iteration is 34.

Test Problem Number 3

After 21th iteration we get the approximate result is as (0,0,0) with the optimal value 0.

Test Problem Number 4



In[8]:=

main[unconcons]

Solve::svars : Equations may not give solutions for all "solve" variables. ≫

feasible solutions are

$\left\{u_1 \to 0,\ x1 \to -\frac{7}{2} - \frac{7\,x3}{2},\ x2 \to \frac{3}{2} + \frac{3\,x3}{2},\ u_2 \to 0,\ u_3 \to 0\right\}$

obj func. val=0

$\left\{x1 \to 1,\ x2 \to 1,\ x3 \to \frac{1}{2},\ u_1 \to \frac{11}{2},\ u_2 \to 6,\ u_3 \to 5\right\}$

obj func. val= $\frac{75}{2}$

$\left\{x1 \to \frac{21}{11},\ x2 \to -\frac{9}{11},\ u_1 \to 0,\ x3 \to -\frac{17}{11},\ u_2 \to 0,\ u_3 \to 0\right\}$

obj func. val=0

$\left\{x2 \to \frac{21}{11},\ x3 \to \frac{3}{11},\ u_1 \to 0,\ x1 \to -\frac{49}{11},\ u_2 \to 0,\ u_3 \to 0\right\}$

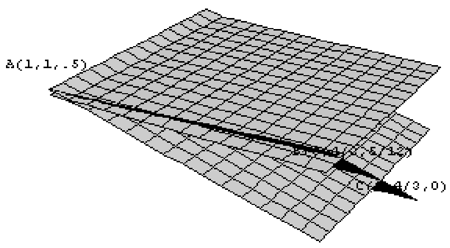
obj func. val=0

optimal

Number of Solution 6

obj func. val= $\frac{75}{2}$

Fig. 2: Graphical representation of TP : 4

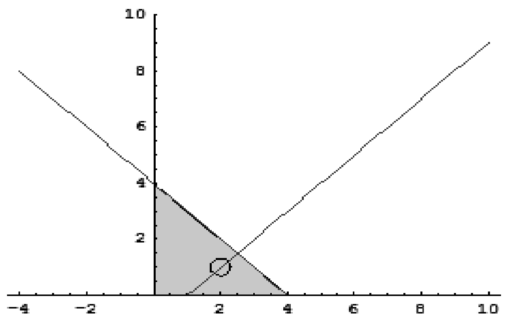


Test Problem Number 7

Outline of Graphical Representation, we have used the following steps to construct the graphical representation of Fig.3 in which way we can visualize the feasible region of the NLP problems.

```
<<Graphics `FilledPlot`
h1=x1+x2+x3+x4-1;
sol=Solve[h1□0,x2]
 {{x2→1-x1-x3-x4}}
g1=sol[[1,1,2]]
 1-x1-x3-x4
{o1,o2}={2,1};
os=0;
f=(-4/3)x1+(-8/3)x2-4x3-10x4+2x1^2+ 4x2^2+6 x3^2 (1/2)x4^2;
Solve[f→os,x2]
       f1=%[[1,1,2]]
fig1=FilledPlot3D[{0,g1},{x1,0,4},{x3,0,4},{x4,0,4},Fills→{{1,2},GrayLevel[.5],GrayLevel[.5],GrayLevel[.5]}]
 FilledPlot3D[{0,1-x1-x3-x4},{x1,0,4},{x3,0,4},{x4,0,4},Fills→{{1,2},GrayLevel[0.5],GrayLevel[0.5],GrayLevel[0.5]}]
m=-1/D[g1,x1]/.x1→o1;Solve[x2-o2→m(x1-o1),x2]/.x1→2;
n1=m(x1-o1)+o2
fig2=Plot[{g1,f1,n1},{x1,4,10},PlotStyle→{RGBColor[1,0,0]},PlotStyle→{RGBColor[0,0,1]},AspectRatio→Automatic]
Show[{fig1,fig2,Graphics[Circle[{o1,o2},.3]]},PlotRange→{0,10},AspectRatio→Automatic]
```

Fig. 3: Graphical representation of TPN: 7



The shaded region and the small circle represent the feasible region and the solution point respectively.

4. Comparison and Discussion

The following representative test problems are used to assess the existing algorithms and our unique computer technique. The analytical properties of these functions can be found in the cited reference.

4.1 Results and Discussions

The efficiency of our technique claimed from the very beginning in this paper is exhibited in the following through numerical experiment with a number of test problems. Here we give a number of numerical examples to show the efficiency of the different techniques.

Example: 1 This numerical example is taken from H. K. Das and Hasan [4].

Consider the functions Maximization: $g = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2$.

Example: 2 This numerical example is taken from H. K. Das, Saha and Hasan [5].

Suppose that the function to be maximized is $f(x) = 62x^2 - 28x - 4$ and choose the initial interval in arbitrarily.

Example: 3 This numerical example is taken from H. K. Das and Hasan [4].

Maximization: $f(x, y, z) = -x^2 - 6y^2 + 3xy + 3yz - z^2$ starting with initial solution (1,1,1).

Example :4 This numerical example is taken from Hasan [10].

Maximization: $Z = (2x_1 + 4x_2 + x_3 + 1)(x_1 + x_2 + 2x_3 + 2)$
Subject-to, $x_1 + 3x_2 \leq 4$, $2x_1 + x_2 \leq 3$, $x_2 + 4x_3 \leq 3$, $x_1, x_2, x_3 \geq 0$

Example: 5 This numerical example is taken from Datta and Hasan [6].

Maximization: $z = 6x_1 + 8x_2 - {x_1}^2 - x_2^2$
Subject to, $4x_1 + 3x_2 = 16$, $3x_1 + 5x_2 = 15$

$x_1, x_2 \geq 0$

Example: 6 This numerical example is taken from Datta and Hasan [6].

Minimization: $z = -x_1 - x_2 + \frac{1}{2} x_1^2 + x_2^2 - x_1x_2$

Subject to, $x_1 + x_2 = 3$, $-2x_1 - 3x_2 = -6$

Example: 7 This numerical example is taken from Datta and Hasan [7].

Maximize $f(x) = 4x_1 + 2x_2 - x_1^2 - x_2^2 - 5$

subject-to, $x_1 + x_2 \leq 4$, $x_1, x_2 \geq 0$

Example: 8 This numerical example is taken from H. K. Das & Hasan [4].

Find the optimal solution to Max $x^2 + 2x$. s/t.$-3 \leq x \leq 5$ with in an initial length of 0.8.

Table 4. Optimality of the Test Problems(TP)

Test Problem Number	Optimality Type	Number of variables	Initial Value	Optimal solution	Optimal value
1	Maximize	2	(0,0)	(1,1)	1
2	Maximize	1	No	0.225807	
3	Maximize	3	No	(0,0,0)	0
4	Maximize	3	No	(1,1,1/2)	75/2
5	Maximize	2	No	$(\frac{35}{11}, \frac{12}{11})$	$\frac{1997}{121}$
6	Minimize	2	No	(3,0)	3/2
7	Minimize	2	No	(2,1)	0
8	Maximize	1	No	*(4.279,5)*	*29.*

In the above Table 4, we have seen that we have taken various types of problems to demonstrate our technique. For this, we had to modify or extended the existing methods.

4.2 Time Comparison

In this section, we present a time comparison chart to show the efficiency of our algorithmic technique. We have used the following computer configuration:

Processor: *Intel(R) Pentium(R) Dual CPU E2180@2.00GHZ 2.00GHZ,* Memory *(RAM):1.00 GB* and the System type: *32-bit operating system*. In briefly, Time Consuming: =TC. The following table has showed that our technique can solve various types of NLP problems. For solving NLP problems with "≥ type" or "= type" constraints, one needs to solve that problem by Two-phase or Big-M simplex method for solving QP problems which is clumsy and time consuming. But in our technique, one does not need to be confused about the type of the NLP and QP problems.

Table 5. Accuracy of our Computer technique

TPN	Existing methods Iteration	Our Iteration	Manual	Coding Time	Command Time
1	35	35	TC	0.21	0.219
2	Failed	19	TC	0.13	0.14
3	21	21	TC	0.221	0.233
4	Algorithmic steps(AS)	AS	TC	0.121	0.243
5	AS	AS	TC	0.102	0.212
6	AS		TC	0.113	0.187
7	AS	AS	TC	0.212	0.243
8	AS	AS	TC	0.50	0.101

We prepare this paper on the basis of existing methods but new work is given with its modification through computer algebra. We use modified phase of Choo & Kim, Golden section, Gradient Search method, Lagrangian multipliers and Khun Tucker Conditions. To our knowledge, there is no code which can solve any NLP problems in a single framework. Finally, we have shown from the above table 5 that our technique is successful than the other existing methods.

5. Conclusion

In this paper, we improved a combined algorithm and developed its computer technique for solving NLP and QP problems. We also focused on the development of the graphical representations of NLP problems. We demonstrated our technique with unconstrained NLP and linearly or non-linearly constrained NLP & QP problems. We observed that the results obtained by our algorithmic technique are completely identical with that of the other methods which are laborious and time consuming.

REFERENCE

[1] Ayoade K., Numerical Experiments with One Dimensional Non-linear Simples search, Computers & Operations Research, **18**, (1991), pp.497-506.

[2] Charnes, A. and W.W. Cooper, Programming with fraction functions, Naval Research Logistics Quarterly, **9**, (1962), pp.181-186.

[3] Choo E. and C. Kim, One dimension simplex search, Computer and Operations Research, **14**, (1987), pp.47-54.

[4] Das H.K. and M. Babul Hasan, A Generalized Computer Technique for Solving Unconstrained Non-Linear Programming Problems**,** Dhaka University Journal of Science, **61(1)**, (2013), pp.75-80.

[5] Das H. K., T. Saha and M. Babul Hasan, A Study on 1-D Simplex Search and its Numerical Experiments through Computer Algebra, Dhaka University Journal of Science, **62(2)**, (2014), pp. 96-102.

[6] Datta B. K. and M Babul Hasan, A computer oriented Lagrange method for Solving Non-Linear Programming Problems, Dhaka University Journal of Scienc*e*, **59(1),** (2011), pp. 71-75.

[7] Datta B. K. and M. Babul Hasan, A Code for Solving Non-Linear Programming Problems, Dhaka University Journal of Science, **59(1)**, (2011), pp. 25-31.

[8] Don, Eugene,Schaum’s Outline Series , McGraw-Hill,New york San Francisco Washington,D.C, (2001).

[9] Jenson D. L. A. J. King, A Decomposition method for QP, IBM SYSREMS JOURNAL, **31(1)**, (1992).

[10] Hasan M. Babul , A Technique for Solving Special Type Quadratic Programming Problems, Dhaka University Journal of Science, **60(2)**, (2012), pp.209-215 .

[11] Sanders J.L, A Nonlinear Decomposition Principle, Operation Research, **13(2)**, (1965), pp.266-271.

[12] Swarup, K., Quadratic Programming” CCERO (Belgium), **8**(2), (1966), pp.132-136.

[13] Wolfram, S. Mathematica, Addision-Wesly Publishing Company, Melno Park, California, New York, (2000).

[14] Wayne L. Winston, Application and algorithms, Dexbury press, Belmont, California, U.S.A, (1994).