Inverse Optimization for Quadratic Programming Problems

SANJAY JAIN[1*] and NITIN ARYA[2]

[1]Department of Mathematical Sciences, Government College, Ajmer
Affiliated to M. D. S. University, Ajmer - 305 001, INDIA
[2]Department of Mathematics, Government Engineering College, Jhalawar
Affiliated to Rajasthan Technical University, Kota, INDIA
e-mails: drjainsanjay@gmail.com; nitin.arya1234@gmail.com

Abstract

In this paper we have proposed a inverse model for quadratic programming (QP) problem in which the parameters in the objective function of the given QP problem are adjusted as little as possible so that the given feasible solution x0 and objective value z0 becomes the optimal ones. We formulate the inverse quadratic programming problem as a linear programming problem having a large number of variables, which can be solved by many existing methods or by the optimization software like TORA, EXCEL SOLVER etc.

Keywords: Inverse optimization, Quadratic Programming, Complementary slackness

1. Introduction

The purpose of this paper is to present a brief taxonomy of an important group of programming problems which occur in certain branches of applied mathematics. A rough heading for these problems is Mathematical Programming Problem. Here special emphasis is given on inverse quadratic programming problem. Recently, there has been interest in inverse optimization problems in the mathematical programming community and a variety of inverse optimization problems have been studied by researchers namely, minimum cost flow problem, NP complete problems, minimum spanning tree problem, shortest path problem, linear programming problem etc.,

Inverse optimization is a relatively new area of research and study of inverse optimization is useful in many branches. In mathematical programming problem, the cost coefficients are not known with precision, it is plausible to call a solution x^0 for minimize cx nearly optimal if there is some nearby cost vector c' such that x^0 is optimal for minimize $c'x$. In numerical analysis, errors for solving the system Ax = b are measured in terms of inverse optimization. Some interesting scenario in which solving the inverse problem is faster than solving for the optimal solution: An example is matroid intersection for

* corresponding author

representable matroids. For some algorithms, the solutions converge to the optimal in an inverse manner: An example is cost scaling algorithms for minimum cost flows.

Ahuja and Orlin [1] provide various references in the area of inverse optimization and compile several applications in network flow problems with unit weight and develop combinatorial proofs of correctness. Zhang and Liu [17] have first been calculated some inverse linear programming problem and further investigated inverse linear programming problems in [18]. Huang and Liu [9] studied on the inverse problem of linear programming problem and its applications. Amin and Emrouznejad [3] has considered applications of inverse problem, Zhang and others [19,10] worked on perturbation approach for inverse linear programming and inverse quadratic programming problem. Scheafer [14] and Wang [16] worked on inverse integer programming problem.

The quadratic programming problem seeks to optimize the objective function of non-negative variables of linear plus quadratic form subject to a set of linear and homogeneous constraints. Zhu [20], Bretthauer, Shetty and Syam [5, 6], Helgason, Kennington and Lall [8], Pardaloes and Kovoor [13], Land and Morton [12], Swarup [15], Chaovalitowangse, Pardalas and Prokopyev [7], Daya and Al-Suitan [4], Al-Khayyal [2], Konno and Kuno [11] and many researchers gave different methods for solving quadratic programming problem.

In the following section, we describe in brief how inverse optimization is applied on QP. In our proposed method, first we obtain the Kuhn Tucker conditions for QP. For obtaining the required optimal solution by inverse optimization, we perturb the coefficients of QP by c to d. Now applying the complementary slackness conditions for the given feasible solution and using standard transformation in inverse QP, it reduces to the linear programming problem (LPP) and the reduced LPP can be solved by many existing methods or by the optimization software like TORA, EXCEL SOLVER etc.

2. Problem Formulation

Let us consider the quadratic programming problem

$$Maximize\ (z) = \sum_{j \in J} c_j x_j + \frac{1}{2}\sum_{j \in J}\sum_{k \in J} c_{jk} x_j x_k$$
$$Subject\ to, \sum_{j \in J} a_{ij} x_j \leq b_i, \text{ for all } i \in I, j \in J$$
$$and\ x_j \geq 0 \text{ for all } j \in J$$

Where I and J are the index sets of decision variables and constraints respectively and $c_{jk} = c_{kj}$ for all j and k.

In inverse optimization, we fix the solution x^0 and optimal value z^0 and for obtaining it, we adjust the parameters in the objective function as little as possible so that x^0 become an optimal solution to the modified QP problem with the objective value z^0. Let d be the adjusted value of c, so the inverse problem is to $minimize\ \|d-c\|$, where $\|.\|$is some selected L_1 norm given by $\|d-c\| = \sum_{j\in J}|d_j - c_j|$. The inverse quadratic programming can be formulated as:

$$\min\ [\sum_{j\in J}(\alpha_j+\beta_j) + \sum_{j\in J}\sum_{k\in J}(\alpha_{jk}+\beta_{jk})]$$

s.t.

$\sum_{k\in F}(\alpha_{jk}-\beta_{jk})\,x^0{}_k - \sum_{i\in B}\lambda_i a_{ij} + \alpha_j - \beta_j + \mu_j = -c_j - \sum_{k\in F} c_{jk}x^0{}_k$ for all $j \in L$
$\sum_{k\in F}(\alpha_{jk}-\beta_{jk})\,x^0{}_k - \sum_{i\in B}\lambda_i a_{ij} + \alpha_j - \beta_j = -c_j - \sum_{k\in F} c_{jk}x^0{}_k$ for all $j \in F$
$\sum_{j\in J}(\alpha_j-\beta_j)\,x^0{}_j + \frac{1}{2}\sum_{j\in J}\sum_{k\in J}(\alpha_{jk}-\beta_{jk})x^0{}_j\,x^0{}_{k_i} =$
$z^* - \sum_{j\in J} c_j x^0{}_j - \frac{1}{2}\sum_{j\in J}\sum_{k\in J} c_{jk}x^0{}_j x^0{}_k$ and $\alpha_{jk},\beta_{jk},\alpha_j,\beta_j \ge 0$ for $j \in J$ $\lambda_i \ge 0, \mu_j \ge 0$ for all $i \in B, j \in J$ where $c_{jk} = c_{kj}$, $d_{jk} = d_{kj}$, $\alpha_{jk} = \alpha_{kj}$ and $\beta_{jk} = \beta_{kj}$ for all j and k

Which is a linear programming problem having large number of variables.

3. Method

Let us consider the quadratic programming problem

$$Maximize\ (z) = f(x) = \sum_{j\in J} c_j x_j + \frac{1}{2}\sum_{j\in J}\sum_{k\in J} c_{jk}x_j x_k$$
$$Subject\ to, \sum_{j\in J} a_{ij}x_j \le b_i, \text{ for all } i \in I, j \in J$$
$$and\ x_j \ge 0 \text{ for all } j \in J \tag{1}$$

Now using surplus/slack variables, we have

$$Maximize\ (z) = f(x) = \sum_{j\in J} c_j x_j + \frac{1}{2}\sum_{j\in J}\sum_{k\in J} c_{jk}x_j x_k$$
$$Subject\ to, \sum_{j\in J} a_{ij}x_j + q_i^2 = b_i, \text{ for all } i \in I, j \in J$$
$$and\ -x_j + r_j^2 = 0 \text{ for all } j \in J \tag{2}$$

Now we construct the Lagrangian function

$$L(x,q,\mu,\lambda,r) = f(x) - [\sum_{i\in I}\lambda_i \sum_{j\in J}(a_{ij}x_j + q_i^2 - b_i) - \sum_{j\in J}\mu_j(-x_j + r_j^2)] \tag{3}$$

If we substitute $q_i^2 = s_i$ than the Kuhn Tucker conditions are as follows:

$$c_j + \sum_{j\in J} c_{jk}x_k - \sum_{i\in I}\lambda_i a_{ij} + \mu_j = 0 \text{ for all } j \in J$$
$$\lambda_i s_i = 0, \mu_j x_j = 0 \text{ for all } i\in I, j \in J \tag{4}$$

The conditions $\lambda_i s_i = 0, \mu_j x_j = 0$ are called complementary slackness conditions for the quadratic programming problem.

These conditions can also be written in the following manner:

If $\sum_{j\epsilon J} a_{ij}x_j < b_i$ then $\lambda_i = 0$
If $x_j > 0$ then $\mu_j = 0$

If x^0 is any feasible solution of quadratic programming and B is the index set of binding constraints in (1) with respect to x^0 (that is $B = \{i: \sum_j a_{ij}x_j{}^0 = b_i\}$, let L and F denote the index set of variables defined as $L = \{j: x_j{}^0 = 0\}$ and $F = \{j: x_j{}^0 > 0\}$, then using these notations the complementary slackness conditions can be restate as:
$\lambda_i = 0$ for all $i \notin B$
$\mu_j = 0$ for all j $\notin L$

We want to make x^0 an optimal solution of the given QP problem by adjusting the cost coefficients in the objective function, so replacing the cost coefficients with d, substituting $x = x^0$ and using the complementary slackness conditions in (4) gives the following characteristics of the cost vector:

$$d_j + \sum_{K\epsilon F} d_{jk}x_k{}^0 - \sum_{i\epsilon B} \lambda_i a_{ij} + \mu_j = 0 \text{ for all } j \epsilon L$$

$$d_j + \sum_{K\epsilon F} d_{jk}x_k{}^0 - \sum_{i\epsilon B} \lambda_i a_{ij} = 0 \text{ for all } j \epsilon F \quad (5)$$

Now substituting $z = z^0$ and $x = x^0$ in the objective function of (1) we have

$$z^0 = \sum_{j \in J} d_j x_j + \tfrac{1}{2}\sum_{j \in J}\sum_{k\in J} d_{jk}x_j{}^o x_k{}^0 \quad (6)$$

Equations (5) and (6) give the characteristics of the new cost vector d. The inverse problem is to perturb the cost vectors c to d so that the feasible solution x^0 becomes an optimal with respect to d such that $\|d - c\|$ is minimum, where $\|.\|$ is some selected L_1 norm given by $\|d - c\| = \sum_{j\epsilon J}|d_j - c_j|$, so the inverse quadratic programming problem can be formulated as:

$$\min[\sum_{j\epsilon J}|d_j - c_j| + \sum_{j\epsilon J}\sum_{K\epsilon J}|d_{jk} - c_{jk}|]$$

s.t.

$$d_j + \sum_{K\epsilon F} d_{jk}x_k{}^0 - \sum_{i\epsilon B} \lambda_i a_{ij} + \mu_j = 0 \text{ for all } j \epsilon L$$
$$d_j + \sum_{K\epsilon F} d_{jk}x_k{}^0 - \sum_{i\epsilon B} \lambda_i a_{ij} = 0 \text{ for all } j \epsilon F$$

$$z^0 = \sum_{j \in J} d_j x_j{}^0 + \tfrac{1}{2}\sum_{j \in J}\sum_{k\in J} d_{jk}x_j{}^o x_k{}^0 \text{ for all } j, k \epsilon F$$
$$\text{and } \lambda_i, \mu_j \geq 0 \text{ and } d_{jk} = d_{kj} \text{ for all } j, k \text{ for all } i \in B, j \in J \quad (7)$$

This is not a linear programming problem but can be converted into a linear programming by using a standard transformation. We know that minimizing $|d_j - c_j|$ and $|d_{jk} - c_{jk}|$are equivalent to minimizing $\alpha_j + \beta_j$ and $\alpha_{jk} + \beta_{jk}$ respectively, subject to the conditions:

$$d_j - c_j = \alpha_j - \beta_{j;}\, p_j, q_j \geq 0$$
$$d_{jk} - c_{jk} = \alpha_{jk} - \beta_{jk;}\, \alpha_{jk}, \beta_{jk} \geq 0 \quad (8)$$

where $d_{jk} = d_{kj}$, $\alpha_{jk} = \alpha_{kj}$ and $\beta_{jk} = \beta_{kj}$ for all j and k.

Using these transformations in (7) the inverse quadratic programming problem becomes a linear programming problem and can be rewrite as

$$\min\left[\sum_{j\in J}(\alpha_j+\beta_j)+\sum_{j\in J}\sum_{K\in J}(\alpha_{jk}+\beta_{jk})\right]$$

s.t.

$c_j+\alpha_j-\beta_j+\sum_{K\in F}(c_{jk}+\alpha_{jk}-\beta_{jk}){x^0}_k-\sum_{i\in B}\lambda_j a_{ij}+\mu_j=0$ for all $j\epsilon L$

$c_j+\alpha_j-\beta_j+\sum_{K\in F}(c_{jk}+\alpha_{jk}-\beta_{jk}){x^0}_k-\sum_{i\in B}\lambda_j a_{ij}=0$ for all $j\epsilon F$

$z^0=\sum_{j\in J}(c_j+\alpha_j-\beta_j){x_j}^0+\frac{1}{2}\sum_{j\in J}\sum_{k\in J}(c_{jk}+\alpha_{jk}-\beta_{jk}){x_j}^o {x_k}^0$ for all $j,k\epsilon F$

and $\alpha_{jk,}\beta_{jk,}\alpha_j,\beta_j,\lambda_i,\mu_j\geq 0$ and $\alpha_{jk}=\alpha_{kj},\beta_{jk}=\beta_{kj}$ for all $i\in B,j,k\in J$ (9)

On further simplifying, these equations can also be written as:

$$\min\left[\sum_{j\in J}(\alpha_j+\beta_j)+\sum_{j\in J}\sum_{K\in J}(\alpha_{jk}+\beta_{jk})\right]$$

s.t.

$\sum_{K\in F}(\alpha_{jk}-\beta_{jk}){x^0}_k-\sum_{i\in B}\lambda_j a_{ij}+\alpha_j-\beta_j+\mu_j=-c_j-\sum_{K\epsilon F}c_{jk}{x^0}_k$ for all $j\epsilon L$

$\sum_{K\in F}(\alpha_{jk}-\beta_{jk}){x^0}_k-\sum_{i\in B}\lambda_j a_{ij}+\alpha_j-\beta_j=-c_j-\sum_{K\epsilon F}c_{jk}{x^0}_k$ for all $j\epsilon F$

$\sum_{j\in J}(\alpha_j-\beta_j){x^0}_j+\frac{1}{2}\sum_{j\in J}\sum_{K\in J}(\alpha_{jk}-\beta_{jk}){x^0}_j{x^0}_k=z^0-\sum_{j\in J}c_j{x^o}_j-\frac{1}{2}\sum_{j\in J}\sum_{k\in J}c_{jk}{x_j}^o{x_k}^0 1$

$\alpha_{jk,}\beta_{jk,}\alpha_j,\beta_j,\lambda_i,\mu_j\geq 0$ for all $i\in B,j\epsilon J$

where $c_{jk}=c_{kj},d_{jk}=d_{kj},\alpha_{jk}=\alpha_{kj},\beta_{jk}=\beta_{kj}$ for all j and k (10)

4. Numerical example

Let us consider a quadratic programming problem

$$\max z=4x_1+6x_2-2{x_1}^2-2x_1x_2-2{x_2}^2$$

s.t.

$$x_1+2x_2\leq 2$$

and

$$x_1,x_2\geq 0$$

It can be written as:

$$\max z=c_1x_1+c_2 6x_2+\frac{1}{2}\sum_{j=1}^{2}\sum_{K=1}^{2}c_{jk}x_{jk}$$

s.t.

$$a_{11}x_1+a_{12}x_2\leq 2$$

and

$$x_1,x_2\geq 0$$

Where $c_1=4,c_2=6,c_{11}=-4,c_{12}=c_{21}=-2,c_{22}=-4,a_{11}=1,a_{12}=2$ and $b_1=2$

${x_1}^*=\frac{1}{3},{x_2}^*=\frac{5}{6}$ is the optimal solution with the objective function value $z^*=4.16$. Let ${x_1}^0=\frac{3}{2},{x_2}^0=\frac{1}{4}$ is a feasible solution of above QP problem and we want to make x^0 an

optimal with the objective function value $z^0 = 4.16$. It can be seen that the constraint is binding with respect to feasible solution x^0 therefore $s_1 = 0$ and also ${x_1}^0 > 0, {x_2}^0 > 0$ 4 therefore by the complementary slackness conditions $\mu_1 = 0$ and $\mu_1 = 0$.
The inverse quadratic programming problem is as follows:

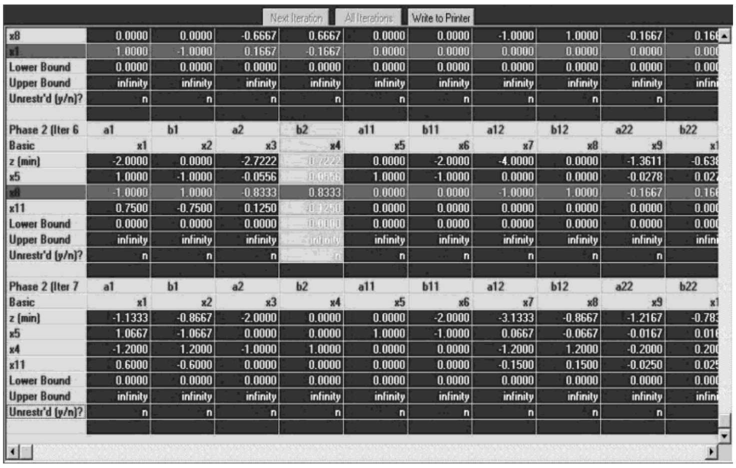
$$\min(\alpha_1 + \beta_1 + \alpha_2 + \beta_2 + \alpha_{11} + \beta_{11} + 2\alpha_{12} + 2\beta_{12} + \alpha_{22} + \beta_{22})$$

s.t.

$$\alpha_1 - \beta_1 + 1.5\alpha_{11} - 1.5\beta_{11} + 0.25\alpha_{12} - 0.25\beta_{12} - \lambda_1 = 2.5$$
$$-\alpha_2 + \beta_2 - 1.5\alpha_{12} + 1.5\beta_{12} - 0.25\alpha_{22} + 0.25\beta_{22} + 2\lambda_1 = 2$$
$$24\alpha_1 - 24\beta_1 + 4\alpha_2 - 4\beta_2 + 18\alpha_{11} - 18\beta_{11} + 6\alpha_{12} - 6\beta_{12} + 0.5\alpha_{22} - 0.5\beta_{22} = 32.66$$

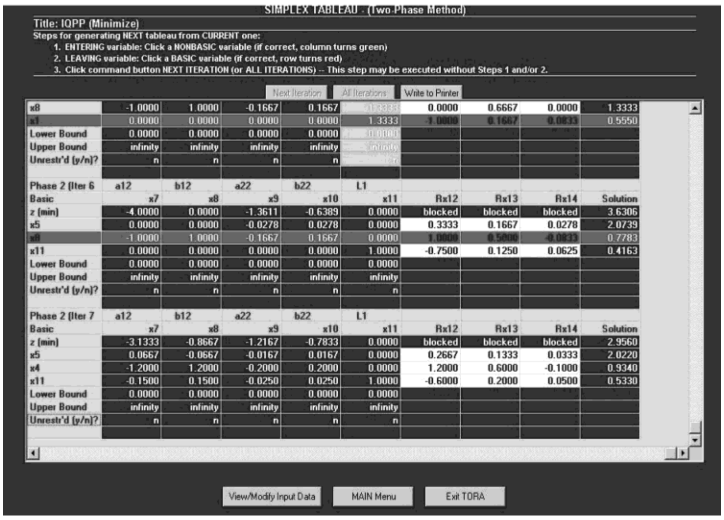
and $\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_{11}, \beta_{11}, \alpha_{12}, \beta_{12}, \alpha_{22}, \beta_{22}, \lambda_1 \geq 0$

Solving this LPP using TORA



Next Iteration | All Iterations | Write to Printer

x8	0.0000	0.0000	-0.6667	0.6667	0.0000	0.0000	-1.0000	1.0000	-0.1667	0.166
x1	1.0000	-1.0000	0.1667	-0.1667	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
Lower Bound	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
Upper Bound	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infini
Unrestr'd (y/n)?	n	n	n	n	n	n	n	n	n	
Phase 2 (Iter 6	a1	b1	a2	b2	a11	b11	a12	b12	a22	b22
Basic	x1	x2	x3	x4	x5	x6	x7	x8	x9	x1
z (min)	-2.0000	0.0000	-2.7222	0.7222	0.0000	-2.0000	-4.0000	0.0000	-1.3611	-0.638
x5	1.0000	-1.0000	-0.0556	0.0556	1.0000	-1.0000	0.0000	0.0000	-0.0278	0.027
x8	-1.0000	1.0000	-0.8333	0.8333	0.0000	0.0000	-1.0000	1.0000	-0.1667	0.166
x11	0.7500	-0.7500	0.1250	0.1250	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
Lower Bound	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
Upper Bound	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infini
Unrestr'd (y/n)?	n	n	n	n	n	n	n	n	n	
Phase 2 (Iter 7	a1	b1	a2	b2	a11	b11	a12	b12	a22	b22
Basic	x1	x2	x3	x4	x5	x6	x7	x8	x9	x1
z (min)	-1.1333	-0.8667	-2.0000	0.0000	0.0000	-2.0000	-3.1333	-0.8667	-1.2167	-0.783
x5	1.0667	-1.0667	0.0000	0.0000	1.0000	-1.0000	0.0667	-0.0667	-0.0167	0.016
x4	-1.2000	1.2000	-1.0000	1.0000	0.0000	0.0000	-1.2000	1.2000	-0.2000	0.200
x11	0.6000	-0.6000	0.0000	0.0000	0.0000	0.0000	-0.1500	0.1500	-0.0250	0.025
Lower Bound	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
Upper Bound	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infinity	infini
Unrestr'd (y/n)?	n	n	n	n	n	n	n	n	n	



SIMPLEX TABLEAU - (Two-Phase Method)

Title: IQPP (Minimize)

Steps for generating NEXT tableau from CURRENT one:

1. ENTERING variable: Click a NONBASIC variable (if correct, column turns green)
2. LEAVING variable: Click a BASIC variable (if correct, row turns red)
3. Click command button NEXT ITERATION (or ALL ITERATIONS) -- This step may be executed without Steps 1 and/or 2.

Next Iteration | All Iterations | Write to Printer

x8	-1.0000	1.0000	-0.1667	0.1667	1.3333	0.0000	0.6667	0.0000	1.3333
x1	0.0000	0.0000	0.0000	0.0000	1.3333	-1.0000	0.1667	0.0833	0.5550
Lower Bound	0.0000	0.0000	0.0000	0.0000	0.0000				
Upper Bound	infinity	infinity	infinity	infinity	infinity				
Unrestr'd (y/n)?	n	n	n	n	n				
Phase 2 (Iter 6	a12	b12	a22	b22	L1				
Basic	x7	x8	x9	x10	x11	Rx12	Rx13	Rx14	Solution
z (min)	-4.0000	0.0000	-1.3611	-0.6389	0.0000	blocked	blocked	blocked	3.6306
x5	0.0000	0.0000	-0.0278	0.0278	0.0000	0.3333	0.1667	0.0278	2.0739
x8	-1.0000	1.0000	-0.1667	0.1667	0.0000	1.0000	0.5000	-0.0833	0.7783
x11	0.0000	0.0000	0.0000	0.0000	1.0000	-0.7500	0.1250	0.0625	0.4163
Lower Bound	0.0000	0.0000	0.0000	0.0000	0.0000				
Upper Bound	infinity	infinity	infinity	infinity	infinity				
Unrestr'd (y/n)?	n	n	n	n	n				
Phase 2 (Iter 7	a12	b12	a22	b22	L1				
Basic	x7	x8	x9	x10	x11	Rx12	Rx13	Rx14	Solution
z (min)	-3.1333	-0.8667	-1.2167	-0.7833	0.0000	blocked	blocked	blocked	2.9560
x5	0.0667	-0.0667	-0.0167	0.0167	0.0000	0.2667	0.1333	0.0333	2.0220
x4	-1.2000	1.2000	-0.2000	0.2000	0.0000	1.2000	0.6000	-0.1000	0.9340
x11	-0.1500	0.1500	-0.0250	0.0250	1.0000	-0.6000	0.2000	0.0500	0.5330
Lower Bound	0.0000	0.0000	0.0000	0.0000	0.0000				
Upper Bound	infinity	infinity	infinity	infinity	infinity				
Unrestr'd (y/n)?	n	n	n	n	n				

View/Modify Input Data | MAIN Menu | Exit TORA we get the solution as

$$\alpha_{11} = 2.0220, \beta_2 = 0.9340, \lambda_1 = 0.5330,$$

Therefore the modified cost coefficients are given as:

$d_{11} = c_{11} + \alpha_{11} - \beta_{11} = -4 + 2.0220 - 0 = -1.9780$

$d_2 = c_2 + \alpha_2 - \beta_2 = 6 + 0 - 0.9340 = 5.0660$

and all other coefficients will remain same.

Using these cost coefficients the modified QP problem is as follows:

$$\max z = 4x_1 + 5.066x_2 - 0.9892{x_1}^2 - 2x_1x_2 - 2{x_2}^2$$

s.t.

$$x_1 + 2x_2 \leq 2$$

and

$$x_1, x_2 \geq 0$$

The optimal solution of modified QP problem is

$$x_1 = 1.5, x_2 = 0.25 \text{ and } z = 4.16$$

5. Particular Case

If we substitute $c_{ij} = 0$ for all i, j in original QP problem, it reduces to linear programming problem and our proposed method reduces for inverse linear programming problem which is studied earlier by [1].

6. Conclusion

Inverse optimization is an important area in both academic research and practical applications. Using the inverse optimization this paper suggested an inversed based methodology for the solution of linear quadratic programming problem. An illustration observation used to demonstrate the advantage of the new approach.

References

[1] Ahuja, R. K., Orlin, J. B., 2001, Inverse Optimization, *Operation Research*, **49**, 771-783.

[2] Al-Khayyal, F. A., 1986, Linear, quadratic and bilinear programming approach to linear complimentary problem, *European Journal of Operational Research*, **24**, 216-227.

[3] Amin, G. R., Emrouznejad, A., 2007, Inverse Forecasting: A New approach for predictive modeling, *Computers & Industrial Engineering*, doi:10.1016/j.cie. 2007.05.007.

[4] Ben-Daya, M. and Al-Sultan, K. S., 1997, A new penalty function algorithm for convex quadratic programming, *European Journal of Operational Research*, **101**, 155-163.

[5] Bretthauer, K. M., Shetty, B. and Syam, S., 1995, A branch and bound algorithm for integer quadratic knapsack problem, *ORSA J. Computing*, **7**, 109-116.

[6] Bretthauer, K. M., Shetty, B. and Syam, S., 1995, A projection method for the integer quadratic knapsack problem, *J. Opl. Res. Soc.*, **47**, 457-462.

[7] Chaovalitowangse, W., Pardalas, P. M. and Prokopyev, O. A.,2004, A new linearization technique for quadratic 0-1 programming problems, *Operation Research Letters*, **32**, 517-522.

[8] Helgason, R., Kennington, J. and Lall, H., 1980, A polynomially bounded algorithm for a singly constrained quadratic programs, *Math. Prog.*, **18**, 338-343.

[9] Huang, S., Liu, Z., 1999, On the inverse problem of linear programming and its application to minimum weight perfect k-matching, *European Journal of Operational Research*, **112**, 421-426.

[10] Jaing, Y., Xiao, X., Zhang, Li. and Zhang, J., 2011, A perturbation approach for a type of inverse linear programming problem. dio: 10.1080/00207160903513003.

[11] Konno, H. and Kuno, T., 1992, Linear Multiplicative Programming, *Mathematical Programming*, 56, 51-64.

[12] Land, A. H. and Morton, G., 1973, An Inverse-Basis Method for Beale's Quadratic Programming Algorithm, *Management Science*, 19, 510-516.

[13] Pardalas, P. M., and Kovoor, N., 1990, An algorithm for a singly constrained class of quadratic program subject to lower and upper bounds, *Math. Prog.*, **46**, 321-328.

[14] Scheafer, A.J., 2009, Inverse integer programming. *Optimization Letters* **3**, 483-489.

[15] Swarup, K., 1966, Quadratic Programming, *CCERO (Belgium)*, **8**, 132-136.

[16] Wang, Li., 2010, Cutting plane algorithm for the inverse mixed integer programming problem. *Operation Research Letters* **37**, 114-116

[17] Zhang, J., Liu, Z., 1996, Calculating some inverse linear programming problems, Journal of Computational and Applied Mathematics. **72**, 261-273.

[18] Zhang, J., Liu, Z., 1999, A further study on inverse linear programming problems, *Journal of Computational and Applied Mathematics*, **106**, 345-359.

[19] Zhang, J., Zhang, Li. And Xiao, X., 2010, A perturbation approach for an inverse quadratic programming problem. dio: 10.1007/s00186-010-0323-4.

[20] Zhu, Z., 2005, An efficient sequential quadratic programming algorithm for nonlinear programming, *Journal of Computational and Applied Mathematics* **175**, 447–464.