Developing an Exact Method for Solving Non Linear Integer Goal Programming Problems

Joni Wilson Sitopu¹, Herman Mawengkang², Saib Suwilo²
¹Graduate School of Mathematics, Universitas Sumatera Utara, Medan, Indonesia
²Department of Mathematics, Universitas Sumatera Utara, Medan, Indonesia

* jwsitopu@gmail.com

Abstract. The nonlinear goal programming problem focused in this paper has a special structure characterized by a subset of variables restricted to assume discrete values, which are linear and separable from the continuous variables. The strategy of releasing nonbasic variables from their bounds, as used in linear goal program, combined with the reduced gradient method, has been developed. This strategy is used to force the appropriate non-integer basic variables within a priority level to move to their neighbourhood integer points.

1. Introducing

Nonlinear integer goal programming problems come up rather certainly in many real life applications. For example, in University Admission Capacity (El-Quliti et al., 2015), allocation of repairable components (Ali et al., 2011), Capacitated transportation problem (Gupta et al., 2013), Software component selection (Kaur and Tomar, 2015), land revitalization strategy (Mayasari et al., 2018). For literature review can be found in (Gur and Eren, 2018).

There are a number of ways to solve the general problem. The algorithms of Chalmet et al. [1], Klein and Hannan [2], and Villareal and Karwan [3] were designed to find all effective solutions. Gabbani and Magazine [4], Gonzales et al [5], Mayasari et al. (2018) have developed interactive methods to solve the problem. Modified Differential Evolution developed by El-Quliti et al. (2015).

Research into multiple objective linear programming programming programming has demonstrated that interactive algorithms are the most exciting solutions to multiple objective programming problems e.g. Evans [17], Steuer [18] and Wierzbicki [19]. In addition, the available interactive problem solving algorithms require an excessive amount of computer resources, both in terms of time requirements and storage space. Some of the algorithms require specialized software for their implementation; some require too much of the decision maker (DM), while others can produce dominated solutions. Thus, an effective and efficient algorithm must now be developed to resolve the problem.

Since integer programming problems are NP-hard, the number of single objective (mixed) integer programming problems that must be resolved to find an acceptable compromise solution must be minimized. The extent to which this goal is achieved by an algorithm can largely determine its applicability/acceptability. It is also desirable to minimize the demands placed on the DM.

We use Analytical Hierarchy Process (AHP) to be used by DM in deciding the priority of the multi objectives in such a way transform the original problem to become mixed integer programming problem. Our aim in this paper is to develop an algorithm that solves only one problem of mixed integer programming for each iteration and does not place too many demands on the DM. The rest of
the paper is arranged as follows: Next, we give the problem statement and some outcomes, then the proposed algorithm is shown and illustrated with a numerical example. We settle the paper with a few comments.

2. Methodology

The nonlinear integer goal programming (NLIGP) problem can be stated mathematically as:

\[
\begin{align*}
\text{Minimize} & \quad (c_1^T p + c_2^T m) \\
\text{subject to} & \quad f(x) + I^T p - I^T m = b \\
\end{align*}
\]

where
\[c_1\ \text{and} \ c_2\ \text{are vectors of weights placed on the violation of constraints.}\]
\[p_i \text{ and } m_i \ \text{are variables showing by how much a given goal is violated.}\]
\[f(x) \ \text{is a real continuous nonlinear function}\]

Note that in a given constraint either \(p_i\) or \(m_i\) is certain to be zero in an optimal solution.

As we knew that goal programming is a problem structure of multi objective programming. Therefore in deriving the method for solving the integer goal programming we start from solving the multi objective nonlinear integer programming (MONLIP).

Consider the MOIP problem:

\[
\begin{align*}
\text{max} & \quad [f_k(x), k \in K] \\
\text{subject to} & \quad x \in X = \{x \mid Ax \leq b, x \in [0, r], x \ \text{integer}\} \\
\end{align*}
\]

where \(K = \{1, 2, \ldots, k\}, A \) is an \(m \times m\) matrix coefficients of the constraints; \(b\) is an \(m\)-vector of the right hand side, \(b \in R^m; f_k(x), k \in K\), is a non linear function of the decision variables, \(x\) is an \(n\)-vector of decision variables, \(x \in R^n\) and \(r\) is an upper bound on \(x\). By maximum, we mean that all objective functions must be simultaneously maximized.

It was our experience that many linear programming problems are excessively large because they try to approximate what is essentially a nonlinear problem by linearizing pieces. Many real-life problems also appear to be such that only a small percentage of the variables are nonlinearly engaged in the objective function. Therefore, we have to consider problems with the following standard form:

\[
\begin{align*}
\text{minimize} & \quad F(x) = f(x^N) + c^T x, \\
\text{subject to} & \quad Ax = b, \\
& \quad l \leq x \leq u \\
\end{align*}
\]

where \(A\) is \(m \times n, m \leq n\). We partition \(x\) into a linear portion \(x^L\) and a nonlinear portion \(x^N\):

\[
x = \begin{bmatrix} x^L \\ x^N \end{bmatrix}
\]

\(x^N\) components are usually referred to as nonlinear variables. Note that all variables \(x\) are operated by \(A\) and \(c\). The part of \(c^T x\) involving \(x^N\) may be incorporated into \(f(x^N)\) in some cases; in other cases, \(c\) may be zero. We assume that function \(f(x^N)\) can be continuously differentiated in the feasible region with gradient

\[
\nabla f(x^N) = g(x^N),
\]

and we assume that both \(f\) and \(g\) can be computed at any feasible point \(x^N\).

The algorithm essentially expands the revised simplex method. It can be characterized as an extension that allows more than \(m\) variables to use some of the associated terminology. Due to the close links with linear programming (LP), we have been able to incorporate many recent advances in LP technology into our implementation. The result is a computer program that has many of the capabilities of an efficient LP code and can handle non-linear terms with the power of a quasi-Newton procedure.
Notation

Partitioning \( x \) and \( F(x) \) in linear and non-linear terms are of considerable practical importance; however, for descriptive purposes, it is convenient to indicate \( F(x) \) and \( V F(x) \) simply by \( f(x) \) and \( g(x) \).

We use upper case letters for matrices, lower case for vectors, and Greek lower case for scalars with some conventional exceptions. The quantity \( \varepsilon > 0 \) symbolizes the accuracy of the floating arithmetic point.

3. Results and Discussion

3.1. Basic solutions; justification for standard form

We need to bring in some linear programming background before we proceed with the nonlinear problem. In particular, equations (1) – (3) with \( f(x) = 0 \) are the standard form for describing and resolving linear programs in the practical implementation of the simplex method. A basic solution is defined by a maximum of \( m \) "basic" variables between other boundaries, while the remaining \( n - m \) "non-basic" variables are equal to one bound. The associated square base matrix \( B \) is drawn from the columns of the constraint matrix \( A \) and the columns \( B \) are replaced one at a time as the simplex method proceeds.

In the standard form, \( A \) and \( x \) is presumed to contain a full matrix of identities and a full set of slack variables. (General equality and inequality constraints are accommodated by placing suitable upper and lower bounds on the slacks.) The standard form is maintained here for many practical reasons. Full justification would involve a lot of implementation background, but very briefly, if columns (but not rows) of \( B \) are changed, a sparse triangular factorization of \( B \) can be maintained more easily. Moreover, although the total number of variables appears to be higher, it is very easy to take advantage of the unit vectors associated with slacks when \( B \) is re-factorized.

3.2. Superbasic variables

The emphasis on the upper and lower limits, \( l \leq x \leq u \), is one of the virtues of the concept of basic solutions. In view of these as "sparse constraints", it is misleading; more importantly, they eliminate a large percentage of the variables. Therefore, the simplex method is free to focus its attention on transforming (factoring) only \( B \) instead of the whole \( A \). (If \( B \) is large and sparse, the problem is enough.)

We cannot foresee an optimal point to be a basic solution to nonlinear problems. It seems reasonable to assume, however, that an optimal solution is "nearly" basic if the number of nonlinear variables is small. Therefore, as a simple generalization, we explain the concept of superbasic variables and the set of general limitations (2) as follows:

\[
Ax = \begin{bmatrix} B & S & N \end{bmatrix} \begin{bmatrix} x_B \\ x_S \\ x_N \end{bmatrix} = b
\]

basics super- nonbasics

basics

Figure 1. Superbasic variables

The matrix \( B \) is square and non-singular, as \( S \) is \( m \times s \) with \( 0 \leq s \leq n-m \) in the simple method, and \( N \) is the remaining columns of \( A \). basics, superbasics and nonbasics are called the associated variables \( x_B, x_S, x_N \). Both basics and superbasics can vary between boundaries. The name is chosen to highlight
the role of superbasics as a "driving force"; they can be moved in any direction (preferably one that improves the objective value), and the basics must be changed in a definite way to maintain the feasibility of the $Ax = b$ constraints.

The main steps to be performed at each iteration are as follows. (They generate a feasible descent direction $p$.)

(A) Compute the reduced gradient $g_A = Z^Tg$  

$$G_A \doteq Z^T GZ$$  

(B) Form some approximation to the reduced Hessian, viz.

$$T_A = GZGZ^T$$  

(C) Obtain an approximate solution to the system of equations

$$Z^T Gp_A = -Z^Tg$$  

by solving the system

$$G_A p_A = -g_A.$$  

(D) Compute the search direction $p = Zp_A$.

(E) Perform a linesearch to find an approximation to $a*$, where

$$f(x + \alpha^* p) = \min_{\alpha \text{ feasible}} f(x + \alpha p)$$  

Apart from having full column rank, eq. (19) is (algebraically) the only constraint on $Z$ and thus $Z$ may take several forms. The particular $Z$ corresponding to our own procedure is of the form

$$Z = \begin{bmatrix} -W & -b^T S \\ I & I \\ 0 & 0 \end{bmatrix} \begin{cases} m \\ s \\ n - m - s \end{cases}$$  

This is a convenient representation, to which we refer in later sections for exposure purposes, but we stress that we only work with $S$ and a triangular (LU) factorization of $B$ on a computer basis. The $Z$ matrix is never calculated.

For many good reasons we consider a $Z$ whose columns are orthonormal ($Z^TZ = I$). The principal advantage is that transformation by such a $Z$ does not introduce unnecessary ill-conditioning into the reduced problem (see steps A through D above, in particular equation (20)). The approach has been implemented in programs, in which $Z$ is stored explicitly as a dense matrix. Extension to large sparse linear constraints would be possible via an $LDV$ factorization of the matrix $[B \ S]$:

$$[B \ S] = [L \ O]DV$$  

where $L$ is triangular, $D$ is diagonal and $D^{1/2}V$ is orthonormal, with $L$ and $V$ being stored in product form. If $S$ has more than one or two columns, however, this factorization is always much denser than the LU factorization of $B$. For reasons of efficiency, we proceed with the $Z$ in (21). At the same time, we know that $B$ must be kept as well as possible (from the unwelcome appearance of $B^{-1}$).

Since (2) is not a linear objective function, no standard algorithm can be used to solve problems with linear or linear integer programming. However, the problem can be introduced as an equivalent of mixed integer linear programming problem. The (2) (or equivalent (B)) solution is a weak efficient (1) solution. However, if an effective solution is wanted, we can solve the following single objective problem:

$$\max_{x \in \lambda} T(x) = \max_{x \in \lambda} \min_{k \in H} \frac{f_k(x) - f_k^*}{f_k^*} + \beta \sum_{k \in K} (f_k(x) - f_k^*)$$  

subject to

$$f_k(x) \geq \overline{f}_k + \alpha(\overline{f}_k - f_k^*), \quad k \in L$$
\[ f_k(x) = \overline{f_k}, \; k \in E \]  
(15)

where $\beta$ is an arbitrary small positive number.

In Theorem 4 (see the Appendix) we prove that the optimal solution of (11) is an efficient solution of (1).

Problem (11) can be reduced to the following equivalent mixed integer linear programming problem.

\[
\max \left( y + \beta \sum_{k \in K} y_k \right)
\]
subject to
\[
\begin{align*}
  f_k(x) - \overline{f_k} & = y_k, \; k \in H \\
  f_k(x) - \overline{f_k} & = -y_k, \; k \in L \\
  f_k(x) - (\overline{f_k} - f_k) y & \geq f_k, \; k \in H \\
  f_k(x) - (\overline{f_k} - f_k) \alpha & \geq \overline{f_k}, \; k \in L \\
  f_k(x) & = \overline{f_k}, \; k \in E \\
  x & \in X \\
  y, y_k & \geq 0, k \in K
\end{align*}
\]
(16) (17) (18) (19) (20) (21) (22) (23)

4. Conclusion
We use Analytical Hierarchy Process (AHP) to be used by DM in deciding the priority of the multi objectives in such a way transform the original problem to become mixed integer programming problem. Our aim in this paper is to develop an algorithm that solves only one problem of mixed integer programming for each iteration and does not place too many demands on the DM. The rest of the paper is arranged as follows: Next, we give the problem statement and some outcomes, then the proposed algorithm is shown and illustrated with a numerical example. We settle the paper with a few comments.

References
[1] G. L. CHALMET, L. LEMONIDIS and D. ELZINGA (1986) An algorithm for the bicriterion integer programming problem. Europ. J. Opl Res. 25, 192-300.
[2] D. KLEIN and E. HANNAN (1982) An algorithm for multi-objective zero-one linear programming problem. Europ. J. Opl Res. 9, 378-385.
[3] B. VILLAREAL and M.H. KARWAN (1981) Multi-criteria integer programming: A (hybrid) dynamic programming recursive approach. Math Prog. 21, 204-223.
[4] D. GABBANI and M.J. MAGAZIE (1986) An iterative heuristic approach for multi-objective programming problems Europ. J. Opl Res. Soc. 37,285-291.
[5] J.J. GONZALEZ, G.R. REEVES and L.S. FRANZ (1993) An interactive procedure for multiple objective integer linear programming problems. In Decision Making with Multiple Objective (Y. Y. HAIMES and V. CHANKONG, Eds). Pp 250-260. Springer Verlag Berlin.
[6] J.N KARAIVANOVA, S.C. NARULA and V. VASSILEV (1993) A interactive procedure for multiple objective integer linear programming problem, Europ. J. Opl Res.(forthoming).
[7] M. H. KARWAN, S. ZIONTS, B. VILLAREAL and R. RAMESH (1995), An improve interactive multicriteria integer programming algorithm. In Decision Making with Multiple Objective (Y. Y. HAIMES and V. CHANKONG, Eds). Pp 261-271. Springer Verlag Berlin.
[8] S. M. LEE and R. MORRIS (1977) Integer goal programming methods. In Decision Making with Multiple Objective (M. K. STARR and M. ZELENY Eds). Pp 273-289. North Holland, New York.
[9] D. MARCOTTE and R. SOLAND (1986) An interactive branch and bound algorithm for multiple criteria optimization Mgmt Sci 32, 61-75.
[10] R. RAMESH, M. H. KARWAN and S. ZIONTS (1989) Preference structure representations using convex cones in multicriteria integer programming. Mgmt Sci 35, 1092-1105.
[11] R. RAMESH, M. H. KARWAN and S. ZIONTS (1986) A class of practical interactive branch and bound algorithm for multi criteria integer programming. Eur. J. Opl. Res. 26, 161-172.
[12] B. VILLAREAL and M. H. KARWAN (1981) An interactive dynamic programming approach to multi criteria discrete programming. J.Math. Anal. Appl. 81, 524-544.
[13] R. RAMESH, M. H. KARWAN and S. ZIONTS (1986) A class of practical interactive branch and bound algorithm for multi criteria integer programming. In Decision Making with Multiple Objective (G. FANDEL and T. FAL, Eds) pp 75-87 Springer Verlag, Berlin.
[14] S. ZIONTS (1977) Integer linear programming with multiple objectives. Ann. Discrete Math. 1, 551-562.
[15] J. TEGHEM and P.L KUNSCH (1985) Interactive methods for multi objective integer linear programming. In Large Scale Modelling and Interactive Decision Analysis (G. FANDEL, M. GRAUER, A. KURZHANSK and A. P. WIERRZBIeki, Eds) pp 75-87 Springer Verlag, Berlin.
[16] J. TEGHEM and P.L KUNSCH (1986) A survey of techniques for finding efficient solutions to multi-objective integer linear programming. Asia Pacific J. Opns. Res. 3, 95-108.
[17] G. W. EVANS (1984) An overview of techniques for solving multi-objective mathematical programs, Mgmt Sci. 30, 1263-1282.
[18] R. STEVER (1986) Multiple Criteria Optimization Theory, Computation and Application, Wiley, New York.
[19] A. WIERZBICKI (1982) A mathematical basis for satisfying decision making. Math Modelling 3, 391-405.
[20] T.L.Saaty., The Analytic Hierarchy Process. McGraw-Hill, New York, 1980 T.L.Saaty., Highlights and critical points in the theory and application of the analytic Hierarchy Process. EJOR 74,3 (1994), 426-447.