International Journal of Mathematics and Computer Research

ISSN: 2320-7167

Volume 13 Issue 10 October 2025, Page no. – 5810-5814

Index Copernicus ICV: 57.55, Impact Factor: 8.615

DOI: 10.47191/ijmcr/v13i10.16

Integer Processing for Mixed Integer Linear Programming Problems

Hardi Tambunan[1], Herman Mawengkang[2]

[1]Department of Mathematics, University of HKBP Nommensen, Medan

[2]Department of Mathematics, University of Sumatera Utara

ARTICLE INFO

Published Online:
24 October 2025

Corresponding Author:
Hardi Tambunan

ABSTRACT

The mixed integer nonlinear programming problem addressed in this paper refers to mathematical programming with continuous and discrete variables and linearities in the objective function and constraints. The purpose of this article is to create an approach to solving MILP problems for integer search through active constraints, and neighborhoods, in order to reduce iterations. The study of the criteria for selecting non-basic variables for use in the integer processes has been conducted using active constraint and neighborhood approaches. The number of integration steps will be limited if the number of integer variables in the problem is finite. However, the integer processes not necessarily depend on the number of integer variables, as many integer variables may have integer values in a continuous optimal solution.

KEYWORDS: Integer, mixed integer linear programming, active constraints, neighborhood

I. INTRODUCTION

Mixed Integer Linear Programming (MILP) refers to mathematical programming with continuous and discrete variables and linearities in the objective function and constraints. There are various applications for the MILP model, including the process industry and the financial, engineering, management science, operations research, and education sectors [1], [2], [3], [4].[5].

The MILP problem can be stated as the following model.

$$\text{Minimize } P = c^T x \quad (1)$$

$$\text{Subject to } Ax \leq b \quad (2)$$

$$x \geq 0$$

$$x_j \text{ integer for some } j \in J$$

Several approaches have been implemented for the integer search process of MILP problems, such as Branch and Bound [6],[7],[8], cutting plane [9]. However, if the objective function and constraints have very large variables, it requires many iterations. The aim of this article is to create an approach for solving MILP problems for integer search through active and neighborhood constraints, so that iterations are fewer.

II. LITERATURE REVIEW

A. Solving Nonlinear Programming Problem

This paper describes our efforts to develop a nonlinear programming algorithm for problems characterized by a large sparse set of linear constraints and a significant degree of nonlinearity in the objective function. It has been our experience that many linear programming problems are inordinately large because they are attempting to approximate, by piecewise linearization, what is essentially a nonlinear problem. Consider problems which have the following standard form:

$$\text{Minimize} \quad F(x) = f(x^N) + c^T x \quad (3)$$

$$\text{Subject to} \quad Ax = b \quad (4)$$

$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \quad (5)$$

where A is $m \times n$, $m \leq n$. We partition $\boldsymbol{x}$ into a linear portion $\boldsymbol{x}^L$ and a nonlinear portion $\boldsymbol{x}^N$:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^N \\ \boldsymbol{x}^L \end{bmatrix} \quad (6)$$

The components of $\boldsymbol{x}^N$ will normally be called the *nonlinear variables.* Note that A and $\boldsymbol{c}$ operate on all variables $\boldsymbol{x}$. In some cases, the part of $\boldsymbol{c}^T\boldsymbol{x}$ involving $\boldsymbol{x}^N$ may be incorporated into $f(\boldsymbol{x}^N)$. In other case $\mathbf{c}$ may be zero. We assume that the function $f(\boldsymbol{x}^N)$ is continuously differentiable in the feasible region, with gradient $\nabla f(\boldsymbol{x}^N) = \boldsymbol{g}(\boldsymbol{x}^N)$, and we assume that both $\boldsymbol{f}$ and $\boldsymbol{g}$ can be computed at any feasible point $\boldsymbol{x}^N$.

In essence the algorithm is an extension of the revised simplex method. To use some of the associated terminology, it might be described as an extension which permits more than m variables to be basic. Partitioning $\boldsymbol{x}$ and $\boldsymbol{F}(\boldsymbol{x})$ into linear and nonlinear terms is of considerable practical

importance. It is convenient to denote $\boldsymbol{F}(\boldsymbol{x})$ and $\nabla\boldsymbol{F}(\boldsymbol{x})$ simply by $f(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$.

Before proceeding to the nonlinear problem, we need to introduce some linear programming background. In particular, equations (3)– (6) with $f(\boldsymbol{x}^N) = 0$ are the standard form for stating linear programs and for solving them in practical implementations of the simplex method. ***A*** *basic solution* is characterized by having at most m "basic" variables lying between their bounds while the remaining $n–m$ "non-basic" variables are equal to one bound or other. An associated square basis matrix $\boldsymbol{B}$ is drawn from the columns of the constraint matrix $\boldsymbol{A}$, and as the simplex method proceeds the columns of $\boldsymbol{B}$ are replaced one at a time [10].

B. Super-basic variables

One virtue of the concept of basic solutions is the emphasis thereby given to the upper and lower bounds, $\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}$. It is misleading to regard these as "sparse constraints". More importantly, they serve directly to eliminate a large percentage of the variables. The simplex method is therefore free to focus its attention on transforming just B, rather than the whole of A. Although a nonlinear problem cannot find an optimal basic solution, if the number of nonlinear variables is small, an optimal basic solution can be obtained. Therefore, as a simple generalization, the concept of super-basic variables can be used and divide the general constraint set (7) as follows:

$$\boldsymbol{A}\boldsymbol{x} = \begin{array}{c} \overset{m}{} \quad \overset{s}{} \quad \overset{n-m-s}{} \\ \left[\begin{array}{ccc} & & \end{array}\right] \\ \text{basics} \quad \text{super-basics} \quad \text{nonbasics} \end{array} \begin{bmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_S \\ \boldsymbol{x}_N \end{bmatrix} = \boldsymbol{b} \qquad (7)$$

The matrix B is square and nonsingular as in the simplex method, S is $m \times s$ with $0 \le s \le n–m$, and N is the remaining columns of A. The associated variables $\boldsymbol{x}_B$, $\boldsymbol{x}_S$, $\boldsymbol{x}_N$ are called the *basics, super-basics* and *non-basics* respectively. Both basics and super-basics are free to vary between their bounds. The name is chosen to highlight the super-basics role as "driving force"; they may be moved in any direction at all, and the basics are then obliged to change in a definite way to maintain feasibility with respect to the constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ [10], [11].

C. Active Constraints

It is assumed that f(x) can be expanded in a Taylor series with second-order remainder

$$f(\boldsymbol{x} + \Delta\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^T\Delta\boldsymbol{x} + \tfrac{1}{2}\Delta\boldsymbol{x}^T G(\boldsymbol{x} + \gamma\Delta\boldsymbol{x})\,\Delta\boldsymbol{x} \qquad (8)$$

where $0 \le \gamma \le 1$, and $G(\boldsymbol{x} + \gamma\Delta\boldsymbol{x})$ is the Hessian matrix of second partial derivatives evaluated at some point between $\boldsymbol{x}$ and $\boldsymbol{x} + \Delta\boldsymbol{x}$. Note that G is a constant matrix if $f(x)$ is a quadratic function.

Partitioned $\Delta\boldsymbol{x}$ and $\boldsymbol{g}(\boldsymbol{x})$ corresponding to the partitioning of A. If $f(x)$ were truly quadratic, we could obtain a constrained stationary point at $\boldsymbol{x} + \Delta\boldsymbol{x}$ by requiring two properties of the step $\Delta\boldsymbol{x}$:

Property 1.

$$\begin{bmatrix} B & S & N \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}_B \\ \Delta\boldsymbol{x}_S \\ \Delta\boldsymbol{x}_N \end{bmatrix} = 0 \qquad (9)$$

i.e., the step remains on the surface given by the intersection of the active constraints.

Property 2.

$$\begin{bmatrix} g_B \\ g_S \\ g_N \end{bmatrix} + G \begin{bmatrix} \Delta\boldsymbol{x}_B \\ \Delta\boldsymbol{x}_S \\ \Delta\boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} B^T & 0 \\ S^T & 0 \\ N^T & I \end{bmatrix} \begin{bmatrix} \mu \\ \lambda \end{bmatrix} \qquad (10)$$

i.e., the gradient at $\boldsymbol{x} + \Delta\boldsymbol{x}$ (given by the left-hand side of Eqn. (10)) is orthogonal to the surface of active constraints and is therefore expressible as a linear combination of the active constraint normal.

For a more general function $f(x)$, the step $\Delta\boldsymbol{x}$ may not lead directly to a stationary point, but we shall use properties 1 and 2 to determine a feasible descent direction.

From (10) obtained:

$$\Delta\boldsymbol{x}_N = 0 \qquad (11)$$

And

$$\Delta\boldsymbol{x}_B = -W\Delta\boldsymbol{x}_s \qquad (12)$$

In which

$$W = B^{-1}S. \qquad (13)$$

Thus,

$$\Delta\boldsymbol{x} = \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} \Delta\boldsymbol{x}_s$$

Equation (11) simplifies when multiplied by the matrix;

$$\begin{bmatrix} I & 0 & 0 \\ -W^T & I & 0 \\ 0 & 0 & I \end{bmatrix} \qquad (14)$$

First it provides an expression for estimates of the Lagrange multipliers for the general constraints:

$$B^T\mu = g_B + \begin{bmatrix} I & 0 & 0 \end{bmatrix} G \begin{bmatrix} -W \\ I \\ 0 \end{bmatrix} \Delta\boldsymbol{x}_s \qquad (15)$$

Note that when $\|\Delta\boldsymbol{x}_s\| = 0$ (which will mean $\boldsymbol{x}$ is stationary). So, it is obtained:

$$B^T\mu = g_B \qquad (16)$$

in which case μ is analogous to the pricing vector π in the revised simplex method. (It is noted that the solution of Eqn. (14) by π.)

Furthermore, it is obtained from (10) that:

$$\boldsymbol{\lambda} = g_N - N^T\mu + \begin{bmatrix}0 & 0 & I\end{bmatrix} G \begin{bmatrix}-W \\ I \\ 0\end{bmatrix} \Delta \boldsymbol{x}_s \quad (17)$$

and again when $\|\Delta \boldsymbol{x}_s\| = 0$ this equation reduces to

$$\boldsymbol{\lambda} = g_N - N^T\pi \quad (18)$$

which is analogous to the vector of reduced costs in linear programming.

The third result from equation (10), following pre-multiplication by the matrix (l0), is an expression for the appropriate step:

$$\begin{bmatrix}-W^T & I & 0\end{bmatrix} G \begin{bmatrix}-W \\ I \\ 0\end{bmatrix} \Delta \boldsymbol{x}_s = -\boldsymbol{h} \quad (19)$$

in which

$$\boldsymbol{h} = \begin{bmatrix}-W^T & I & 0\end{bmatrix} \boldsymbol{g} = \boldsymbol{g}_s - W^T \boldsymbol{g}_B = \boldsymbol{g}_s - S^T \boldsymbol{\pi} \quad (20)$$

The form of equation (19) suggests that

$$\begin{bmatrix}-W^T & I & 0\end{bmatrix} G \begin{bmatrix}-W \\ I \\ 0\end{bmatrix} \quad (21)$$

can be regarded as a "reduced" Hessian and $\boldsymbol{h} = \begin{bmatrix}-W^T & I & 0\end{bmatrix} \boldsymbol{g}$ a reduced gradient, with (20) giving a Newton step in the independent variables $\Delta \boldsymbol{x}_s$. Note that $\|\boldsymbol{h}\| = 0$ becomes a necessary condition for a stationary point on the current set of active constraints, which, if the reduced Hessian is nonsingular, implies that $\|\Delta \boldsymbol{x}_s\| = 0$. So that a class of algorithms is obtained, where the search direction along the active constraint surface is characterized as being in the range of the Z matrix which is orthogonal to the normal constraint matrix. Thus, if $\hat{A}\boldsymbol{x} = \hat{\boldsymbol{b}}$ is the current set of $n - s$ active constraints, Z is an $n \times s$ matrix such that $\hat{A}Z = 0$ [8-10],[9-11].

D. Neighbourhood Search

The test is conducted by a search through the neighbours of a proposed feasible point to see whether a nearby point is also feasible and yields an improvement to the objective function, Scarf [18] has proposed a quantity test to replace the pricing test for optimality in the integer programming problem. Let $[\beta]_k$ be an integer point belongs to a finite set of neighbourhood $N([\beta]_k)$. Define a neighbourhood system associated with $[\beta]_k$, that is, if such an integer point satisfies the following two requirements:

If $[\beta]_j \in N([\beta]_k)$ then $[\beta]_k \in [\beta]_j), j \neq k$, and

$N([\beta]_k) = [\beta]_k + N(0)$.

The proposed integration strategy can be described as follows. Given a non-integer component, x_k, of an optimal vector , x_B. The adjacent points of x_k, being considered are $[x_k]$ and $[x_k] + 1$. If one of these points satisfies the constraints and yields a minimum deterioration of the optimal objective value we move to another component, if not we have integer-feasible solution. Let $[x_k]$ be the integer feasible point which satisfies the above conditions. We could then say if $[x_k] + 1 \in N([x_k])$ implies that the point $[x_k] + 1$ is either infeasible or yields an inferior value to the objective function obtained with respect to $[x_k]$. In this case $[x_k]$ is said to be an "optimal" integer feasible solution to the integer programming problem [12], [13].

III. METHODOLOGY

A. Integer Processing

Consider a Mixed Integer Linear Programming (MILP) problem with the following form;

$$\text{Minimize } P = c^T x \quad (22)$$

$$\text{Subject to } Ax \leq b \quad (23)$$

$$x_j \geq 0$$

$$x_j \text{ integer for some } j \in J \quad (24)$$

A component of the optimal basic feasible vector $(x_B)_k$, to MILP solved as continuous can be written as

$$(x_B)_k = \beta_k - \alpha_{k1}(x_N)_1 - \cdots - \alpha_{kj}(x_N)_j - \cdots - \alpha_{kn} - m(x_N)_N n - m \quad (25)$$

If $(x_B)_k$ is an integer variable and we assume that β_k is not an integer, the partitioning of β_k into the integer and fractional components is that given;

$$\beta_k = \lfloor \beta_k \rfloor + f_k, 0 \leq f_k \leq 1 \quad (26)$$

Suppose to increase $(x_B)_k$ to its nearest integer, $([\beta] + 1)$. Based on the idea of suboptimal solutions may elevate a particular non-basic variable, say $(x_N)_{j^*}$, above its bound of zero, provided α_{kj^*}, as one of the elements of the vector α_{j^*}, is negative. Let Δ_{j^*} be amount of movement of the non-variable $(x_N)_{j^*}$, such that the numerical value of scalar $(x_B)_k$ is integer. Referring to Eqn. (25), Δ_{j^*} can then be expressed as

$$\Delta_{j^*} = \frac{1 - f_k}{-\alpha_{kj^*}} \quad (27)$$

while the remaining non-basic stay at zero. It can be seen that after substituting (26) into (27) for $(x_N)_{j^*}$ and taking into account the partitioning of β_k given in (26), obtained;

$$(x_B)_k = [\beta] + 1 \quad (28)$$

Thus, $(x_B)_k$ is an integer.

It is now clear that a non-basic variable plays an important role to integration process the corresponding basic variable. However, if the result is non-integer, the following integer process is carried out again.

B. Pivoting

The following result is necessary in order to confirm that must be a non-integer variable to work with in integer processes. Another component, $(x_B)_{i\neq k}$, of vector x_B will also be affected as the numerical value of the scalar $(x_N)_{j^*}$ increases to Δ_{j^*}. Consequently, if some element of vector α_{j^*}, i.e., α_{j^*} for $i \neq k$, are positive, then the corresponding element of x_B will decrease, and eventually may pass through zero. However, any component of vector x must not go below zero due to the non-negativity restriction. Therefore, a formula, called the minimum ratio test is needed in order to see what is the maximum movement of the non-basic $(x_N)_{j^*}$ such that all components of x remain feasible. This ratio test would include two cases.

1. A basic variable $(x_B)_{i\neq k}$ decreases to zero (lower bound) first.
2. The basic variable, $(x_B)_k$ increases to an integer.

Specifically, corresponding to each of these two cases above, one would compute;

$$\theta_1 = \min_{i\neq k|\alpha_{j^*}>0}\left\{\frac{\beta_i}{\alpha_{j^*}}\right\} \quad (29)$$

$$\theta_2 = \Delta_{j^*} \quad (30)$$

How far one can release the non-basic $(x_N)_{j^*}$ from its bound of zero, such that vector x remains feasible, will depend on the ratio test θ^* given below;

$$\theta^* = \min(\theta_1, \theta_2) \quad (31)$$

obviously, if $\theta^* = \theta_1$, one of the basic variables $(x_B)_{i\neq k}$ will hit the lower bound before $(x_B)_k$ becomes integer. If $\theta^* = \theta_2$, the numerical value of the basic variable $(x_B)_k$ will be integer and feasibility is still maintained. Analogously, we would be able to reduce the numerical value of the basic variable $(x_B)_k$ to its closest integer $[\beta_k]$. In this case the amount of movement of a particular non-basic variable, $(x_N)_{j^*}$, corresponding to any positive element of vector $\alpha_{j'}$, is given by

$$\Delta_{j'} = \frac{f_k}{\alpha_{kj}} \quad (32)$$

In order to maintain the feasibility, the ratio test θ^* is still needed. Consider the movement of a particular non-basic variable, Δ, as expressed in Eqns. (28) and (32).

The only factor that one needs to calculate is the corresponding element of vector α. A vector α_j can be expressed as

$$\alpha_j = B^{-1}a_j, j = 1 \ldots, n-m \quad (33)$$

Therefore, in order to get a particular element of vector α_j we should be able to distinguish the corresponding column of matrix $[B]^{-1}$. Suppose we need the value of element α_{kj^*}, letting v_k^T be the k-th column vector of $[B]^{-1}$, we then have

$$v_k^T = e_k^T B^{-1} \quad (34)$$

Subsequently, the numerical value of α_{kj^*} can be obtained from

$$\alpha_{kj^*} = v_k^T a_{j^*} \quad (35)$$

in Linear Programming (LP) terminology the operation conducted in Eqns. (34) and (35) is called the pricing operation. The vector of reduced costs d_j is used to measure the deterioration of the objective function value caused by releasing a non-basic variable from its bound. Consequently, in deciding which non-basic should be released in the integration process, the vector d_j must be taken into account, such that deterioration is minimized. Recall that the minimum continuous solution provides a lower bound to any integer-feasible solution. Nevertheless, the amount of movement of particular non-basic variable as given in Eqns. (31) or (32), depends in some way on the corresponding element of vector α_j. Therefore, it can be observed that the deterioration of the objective function value due to releasing a non-basic variable $(x_N)_{j^*}$ so as to integer a basic variable $(x_B)_k$ may be measured by the ration

$$\left|\frac{d_k}{\alpha_{kj^*}}\right| \quad (36)$$

where $|a|$ means the absolute value of scalar a.

In order to minimize the detonation of the optimal continuous solution we then use the following strategy for deciding which non-basic variable may be increased from its bound of zero, that is,

$$\min_j\left\{\left|\frac{d_k}{\alpha_{kj^*}}\right|\right\}, \quad j = 1, \ldots, n-m \quad (37)$$

From the "active constraint" strategy and the partitioning of the constraints corresponding to basic(B), super-basic (S) and non-basic (N) variables we can write

$$\begin{bmatrix} B & S & N \\ 0 & 0 & I \end{bmatrix}\begin{bmatrix} x_b \\ x_N \\ x_S \end{bmatrix} = \begin{bmatrix} b \\ b_N \end{bmatrix} \quad (38)$$

$$Bx_b + Sx_N + Nx_S = b \quad (39)$$

or

$$x_N = b_N \quad (40)$$

The basis matrix B is assumed to be square and nonsingular, So, it is obtained;

$$x_B = \beta - Wx_S - \alpha x_N \quad (41)$$

in which

$$\beta = B^{-1}b \quad (42)$$

$$W = B^{-1}S \quad (43)$$

$$\alpha = B^{-1}N \quad (44)$$

Expression (44) indicates that the non-basic variables are being held equal to their bound. It is evident through the "nearly" basic expression of Eqn. (41). Particularly, we would be able to release a non-basic variable from its bound, Eqn. (44) and exchange it with a corresponding basic variable in the integration process, although the solution would be degenerate.

In a position where particular basic variable, $(x_B)_k$ is being integer, thereby a corresponding non-basic variable, $(c_N)_{j^*}$, is being released from its bound of zero. Suppose the maximum movement of $(x_N)_{j^*}$ satisfies $\theta^* = \Delta_{j^*}$ such that $(x_B)_k$ is integer valued to exploit the manner of changing the basis in linear programming, we would be able to move $(x_N)_{j^*}$ into B (to replace $(x_B)_k$) and integer-valued $(x_B)_k$ into S in order to maintain the integer solution. We now have a degenerate solution since a basic variable is at its bound. The integration process continues with a new set $[B, S]$. In this case, eventually we may end up with all of the integer variables being super-basic. The other case which can happen is that a different basic variable $(x_B)_{i\neq k}$ may hit its bound before $(x_B)_k$ becomes integer [14]

IV. CONCLUSIONS

This paper presents an integer search process for achieving integer feasibility in a class of mixed integer nonlinear programming problems in a relatively short time. The search approach uses a strategy of releasing non-basic variables from their boundaries, combined with the active constraint method and the concept of super-basic variables. After solving the problem by ignoring the integrality requirement, this strategy is used to force the corresponding non-integer basis variables to move to integer points in their neighborhood.

Studies on the criteria for selecting non-basic variables for use in the integration process have also been conducted. The number of integration steps will be limited if the number of integer variables in the problem is finite. However, it should be noted that the computation time for the integration process does not necessarily depend on the number of integer variables, as many integer variables may have integer values in the continuous optimal solution.

REFERENCES

1. Grossmann and, I.E., Sahinidis, N.V (eds). 2002. *Special Issue on Mixed-integer Programming and its Application to Engineering*, Part I, *Optim. Eng., 3 (4)*, Kluwer Academic Publishers, Netherlands
2. Tambunan, H., and Mawengkang, H. 2018. Integer Linear Programming Approach for Detection Learning Outcomes Achievement. *Far East Journal of Mathematical and Sciences*, 5(1), 95-109, 2018
3. Tambunan, H. 2017. Designing Multimedia Learning for Solving Linear Programming. *Global Journal of Pure and Applied Mathematics*, 12 (6). 5265-5281.
4. Tambunan, H. 2025.The Role of Multimedia for Solving Integer Linear Programming. *International Journal of Mathematics and Computer Research*, 13(6), 5342-5345.
5. Tambunan, H. 2022. A Mathematical Modeling for Education. *AIP Conference Proceedings*, 2577,020068-1-020068-6
6. Gupta, O.K; and Ravindran, V. 1985. Branch and Bound Experiments in Convex Nonlinear Integer Programming. *Management Science*, 31, 1533-1546.
7. Belotti, P., Lee, J, Liberti, L., Margot, F., and Wachter, A. 2009. Branching and Bounds Tightening Techniques for non-Convex MINLP, *Optimization Methods and Software,* 24(4**)**:597-634.
8. Westerlund, T., and Petersson, F. 1995. A Cutting Plane Method for Solving Convex MINLP Problems, *Computers Chem. Eng*., 19, 131–136
9. Murtagh, B.A., and Sargent, R.W.H. 1969. A constrained minimization method with quadratic convergence, in: R. Fletcher, ed., *Optimization* (Academic Press, New York, 215-246.
10. Mawengkang, H., and Murtagh, B. A. 1986. Solving Nonlinear Integer Programs with Large-Scale Optimization Software. *Annals of Operations Research 5425-437*
11. Scarf, H. E. 1986. *Testing for Optimality in the Absence of Convexity*. In: Heller, W.P., Starr, R. M., and Starett, D. A (Eds) Cambridge University Press, 117-134
12. Tambunan, H., and Mawengkang, H. 2025. Neighbourhood Approach for Solving One Class of Mixed Integer Non-Linear Programming. *International Journal of Mathematics and Computer Research*, 13(1), 4727-4730.
13. Tambunan, H., and Mawengkang, H. 2016. Solving Mixed Integer Non-Linear Programming Using Active Constraint. *Global Journal of Pure and Applied Mathematics*, 13(7), 2965-2973.