Journal of the Nigerian Association of Mathematical Physics
Volume **19 (November, 2011), pp 475 – 482**
© J. of NAMP

Summary Of Branch And Bound Algorithm For BIP (0,1) Using LP Relaxation

[1] Abdullahi, N and [2] Sani, B

[1]Department of Mathematics and Computer Science,
NDA, Kaduna
[2]Department of Mathematics,
Ahmadu Bello University Zaria

Abstract

In this paper we contemplate the Binary integer programming problems and presents a summary of the techniques of finding an optimal solution, using the Branch-and-Bound algorithm highlighting some critical features and new trend in published articles not yet commonly found in most text books. A generic algorithm is presented and illustrated by solving a knapsack problem.

Keywords: Linear programming, Binary Integer programming, Relaxation, Branch and Bound, Variable selection, Partitioning, fathoming, Depth-first search, Fixing Variable.

1.0 Introduction

In this paper we are dealing with a special type of Integer programming (IP) problem, the class of (0-1) integer program called Binary Integer Program (BIP). An IP problem is a Linear programming problem (LPP), in which all of the decision variables are restricted to take on only integer values. If some (not all) of the variables are restricted to take integral values then the optimization problem is called Mixed integer program (MIP).The situation where the variables are restricted to take only 0-1 values is called BIP. These are problems of the following types;

$$\begin{array}{c} Maximize \quad C^T x \\ Subject\ to \quad Ax \le b \\ x_j \in \mathbb{Z}_+^n,\ j = 1, \ldots\ldots p \end{array} \qquad (1)$$

where $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, C \in \mathbb{R}^n, x \in \mathbb{R}^n\ and\ we\ assume\ n = p\ and\ x \in \{0,1\}$

Among the most successful methods of solving (1) currently are linear programming (LP) based on branch-and-bound (B&B) algorithm, which is an enumerative technique. Most commercial LP solvers are based on this method. Among the classical B&B based methods for solving general IPP and its variants including (1) are the Branch-and-cut (B&C), Branch-and Price (B&P) and B&C with interior point method [1].

B&C method consist of a cutting plane method embedded within B&B algorithm, while B&P is a generalization of LP based B&B specifically designed to handle IP formulations that contain a huge number of variables. It is a combination of B&B with column generation. B&C with interior point method is the construction of B&C algorithm of the usual type, but then uses interior point method to solve the LP relaxations [11].

As we will see below, B&B algorithms leave two choices: how to split a problem (branching) and which (sub) problem to select next. In this paper we focus on the B&B algorithm for the class of 0-1 IPP and give a summary of the major steps with a numerical example and some recent trend published in the literature.

In section 2 we review the literature on the subject. In section 3 we present B&B subroutines for 0-1 IP, a numerical example is given, and a generic B&B algorithm for 0-1 IP is presented.

Corresponding authors: E-mail: nasirualbani@yahoo.co.uk, dbsani@yahoo.co.uk, Tel. +2348035587641(Abdullahi)

2. Branch and Bound Review

The B&B technique is currently the most widely used method for solving IP and MIP problems. The method consists of four basic steps; initialization, branching, bounding, and fathoming. In the initialization step, the associated LP problem is solved. If the solution obtained is integer, then the IP problem has been solved; otherwise the solution forms what is called the "starting node." The branching step partitions this set of solutions into several subsets. Partitioning is a recursive process. The bounding step obtains a bound for each new subset; this bound is actually the value of the objective function of the LP problem of a subset. In the fathoming step, a decision is made at the end of each branch (node) either to exclude some subsets from further consideration (fathom them) or to continue partitioning. The tree enumeration procedure ends when all possible branches or subsets are fathomed.

The B&B concept was introduced by [8]. Later [4] modified the Land and Doig algorithm by using inequalities instead of equalities for the branching step at each unfathomed node. Mitten [10] gives a general description of B&B approaches. The special technique for solving 0-1 IP problem has been the Implicit Enumeration algorithm due to [3].

2.1 Implicit Enumeration

Implicit enumeration algorithms are motivated by the fact that the integer solution space of combinatorial problems consists of a finite number of points. Accordingly, the approach seeks to examine explicitly only some solutions and implicitly rule out the rest of the solutions as being infeasible or sub-optimal. It also uses a tree enumeration procedure and implements the basic concept of branching, bounding, and fathoming of B&B methods. The most useful type of implicit enumeration technique is applied to the special class of 0-1 (binary) integer problems.

Balas was the first to propose the use of implicit enumeration to solve binary problems. The arithmetic operations of his "additive algorithm" involve only addition and subtraction. Glover [6, 7] developed a "surrogate constraint" that combines all or some of the original constraints of the problem without eliminating any of the original feasible integer points. Thus, it enforces restrictions upon the optimal solution that cannot be determined from any of the individual constraints. Geoffrion [5] used an imbedded LP to compute surrogate constraints slightly different from those developed by Glover. His approach synthesizes Balasian implicit enumeration with the approach typified by [8].

Computational experience indicates that the solution time of binary implicit enumeration varies exponentially with the number of variables. Moreover, the prospect of generalizing the 0-1 additive approach to all-integer problems is not promising.

2.2 A general B&B algorithm for IPP.

A formal statement of a general B&B algorithm is presented below. We use the following notation: Let S denote the constraint set of (1) i.e $S = \{x \in \mathbb{Z}_+^n = \{0,1\}: Ax \le b\}$ and S_R the corresponding continuous relaxation set of (1) i.e $S_R = \{x \in \mathbb{R}^n: Ax \le b\}$, where $S \subseteq S_R$ and $C_x^T \le C_R^T \; \forall x \in S$. Clearly solving a problem relaxation (ie dropping the integrality restriction) provides an upper bound on the objective value of the underlying problem. Perhaps the most common relaxation of (1) is the LP relaxation formed by dropping the integer restrictions and enforcing appropriate bound condition on the variables. There are other forms of relaxations (like Lagrangean relaxation etc) for details see [1].

The notation L is used to denote the list of active subproblems $\{IP^i\}$ where IP^0 = IP denotes the original IP. The notation $\overline{Z}_i$ denote an upper bound on the optimal objective value of IP^i, and $\underline{Z}_{ip}$ denotes the incumbent objective value.(i.e the objective value corresponding to the current best integral feasible solution to IPP).

2.2.1 General B&B algorithm

1 (Initialization): Set $L = \{IP^0\}, \bar{z}_0 = +\infty$ and $\underline{Z}_{ip} = -\infty$

2 (Termination): If L = $\emptyset$, then the solution x^* exists (i.e $\underline{Z}_{ip} = -\infty$) then IP is infeasible.

3 (Problem selection and relaxation): Select and delete a problem IP^i. Let z_i^R denote the optimal objective value of the relaxation, and let x^{iR} be an optimal solution if one exists.(Thus $z_i^R = c^T x^{iR}, or\ z_i^R = -\infty$).

4 (fathoming and Pruning):

(a) If $z_i^R \le \underline{Z}_{ip}$ go to step 2

(b) If $z_i^R > \underline{Z}_{ip}$ and x^{iR} is integral feasible, update $\underline{Z}_{ip} = z_i^R$ .Delete from L all problems with $Z_i \le \underline{Z}_{ip}$. Go to step 2.

5 (Partitioning): Let $\{S^{ij}\}_{j=1}^{j=k}$ be a partition of the constraint set S^i of problem IP^i . Add problems $\{IP^{ij}\}_{j=1}^{j=k}$ to L, where IP^{ij} is IP^i with feasible region restricted to S^{ij} and $\bar{z}_{ij} = z_i^R$ for j = 1,…..,k. Go to step 2. [13].

The actual implementation of a B&B algorithm is typically viewed as a tree search, where the problem at the root node of the tree is the original IP. The tree is constructed in an iterative fashion with new nodes formed by branching on an existing node for which the optimal solution of the relaxation is fractional.

To simplify the notation a bit, let node i in the search tree be denoted by v_i with the root node given by v_0 For a path P_i in the tree connecting v_0 to v_i, let S^i be the intersection of S with the set of points satisfying the constraints given by the edges of P_i. If P_i has k+1 nodes ordered as

$$v_0 = v_{i(0)}, v_{i(1)}, \dots\dots\dots v_{i(k-1)}, v_{i(k)} = v_i$$

Then $S = S^{i(0)} \supseteq S^{i(1)} \supseteq \cdots \quad \supseteq S^{i(k)} = S^i$

Now suppose that the enumeration is at node i in the tree. The problem considered at v_i is

$$\underline{Z}_{\text{ip}} = \max\{\text{cx} : \text{x} \in S^i\} \tag{2}$$

An upper bound $\underline{Z}_i \geq \underline{Z}_{\text{ip}}$ may be calculated by considering the relaxation of (2)

$$\underline{Z}_i = Z_i^R = Max\{C^T x : x \in S_R^i \supseteq S^i\} \tag{3}$$

which we call LP^i. Note that an upper bound at a node is valid for any of its Successor; i.e if v_k is a successor of v_i, then $S_R^i \supseteq S^i \supseteq S^k$. For the IP we have:

$$S^i = \{x : A^i x = b^i, x \geq 0 \; and \; integer\} \tag{4}$$

$$S_R^i = \{x : A^i x = b^i, x \geq 0 \} \tag{5}$$

So $\underline{Z}_i$ is calculated by solving the corresponding LP.

2.2.2 Partition

Suppose that the LP (3) is solved at v$_i$ and the solution x^{iR} is not all integer; in particular some basic variables $x_{Bj} = \lfloor \overline{b_j} \rfloor + \theta_j, \; 0 < \theta_j < 1$, then a partition of S^i is

$$\{S^i \cap \{x : x_{Bj} \leq \lfloor \overline{b_j} \rfloor\}, S^i \cap \{x : x_{Bj} \geq \lceil \overline{b_j} \rceil\}\} \tag{6}$$

Where $\lceil \varphi \rceil$ denotes the smallest integer greater than or equal to φ , and $\lfloor \varphi \rfloor$ denotes the largest integer less than or equal to φ . Now if we assume S is bounded and

let

$$S^i = \{x : Ax = b, 0 \leq \alpha_j^i \leq x_j \leq \beta_j^i \leq u_j, x_j \; integer, j = 1, .., n\} \tag{7}$$

At node v_i, where α_j^i and β_j^i are integers determined from (5), and $\alpha_j^0 = 0$ and $\beta_j^0 = u_j$.
To perform the computations, a branching strategy is needed at step 4. Lee and Mitchel [9] presents some branching rules found in the literature. While [2] revisits the branching rules in respect of Constraint Integer Programming applicable to both MIP and 0-1 IP. A variable selection strategy is needed in step 5. Also [9] provide some useful strategies commonly use in practice.

3 B&B Subroutines for 0-1 IP

Although most of the steps of a B&B algorithm are general in that they are appropriate for a variety of problem classes, several computational procedures are problem dependent. The five routines below are used to guide the search for the optimum and to extract information that can be used to reduce the size of the B&B tree.

Bound: This procedure examine the relaxed problem at a particular node and tries to establish a bound on the optimum. It has two possible outcomes:

(i) An indication that there is no feasible solution in the set of integer solutions represented by the node, or
(ii) A value $\bar{z}_i = z_{UB}$, an upper bound on the objective function for all solutions at node i, and its descendent nodes.

Approximate: This procedure attempts to find a feasible integer solution from the solution of the relaxed problem. If one is found it will have an objective value, call it $\underline{z}_i = z_{LB}$,that is a lower bound on the optimum for a maximization problem.

Variable Fixing: This procedure performs logical tests on the solution found at node. The goal is to determine if any of the free binary variables are necessarily 0 or 1 in an optimal integer solution at the current node or at its descendants, or whether they must be set to 0 or 1 to assure feasibility as the computations progress.
Branch: A procedure aimed at selecting one of the free variables for separation. Also decided is the first direction (0 or 1) to explore.
Backtrack: This is primarily a bookkeeping procedure that determines which node to explore next when the current node is fathomed. It is designed to systematically enumerate all remaining live nodes of the B&B tree while assuring that the optimal solution to the original IP is not overlooked.
Example: 0-1 knapsack problem

$$Maximize \quad Z = 4x_1 + 9x_2 + 6x_3$$
$$subjet\ to \quad 5x_1 + 8x_2 + 6x_3 \leq 12$$
$$x_j = 0\ or\ 1, \quad j = 1,2,3$$

The B&B tree of this example is shown in fig. 1. The order in which the variables appear is determined by the “bang for buck” (c_j/a_j, objective coefficient/constraint coefficient) rule. The one with the highest ratio is the best item (variable) to place in the knapsack. This is designed to provide tight upper bounds and increase the likelihood of fathoming (see Table 1). It also speeds convergence. [12].

Table 1. Bang for buck table

Variable, j	1	2	3
Benefit/cost, C_j/a_j	0.8	1.25	1.0

3.1 Variable Fixing and Data structure for 0-1 IP

Given an optimal LP solution to a maximization problem, the reduced costs $\overline{c}_j$, are nonnegative for all non basic variables x_j at their lower bound (typically zero), and non positive for all non basic variables at their upper bound. This leads to the following result which is valid for any MIP.

Proposition: Let $x \in \mathbb{R}^n$ be the decision variables in an IP such that $x_j \geq 0\ \forall\ j, and\ let\ z_R^i\ and\ \underline{z}_{ip}$ be the objective values of the LP relaxation and incumbent, respectively. If x_j is non basic at its lower bound (zero) in the solution to the LP relaxation, and $z_R^i - \bar{c}_j \leq \underline{z}_{ip}$, then there exists an optimal solution to the integer program with x_j at its lower bound. Similarly, if x_j is non basic at its upper bound in the solution to the LP relaxation, and $z_R^i + \bar{c}_j \leq \underline{z}_{ip}$, then there exists an optimal solution to the integer program with x_j at its upper bound. [14].

Branching to the left first ($x_j = 1$): For each node k, there is a path P_k leading to it from node 0 which corresponds to an assignment of binary values to a subset of the variables. Such an assignment is called a *partial solution*. We denote the index set of assigned variables by $W_k \subseteq N = \{1, \dots, n\}$ and let

$$S_k^+ = \{j : j \in W_k\ and\ x_j = 1\}$$

$$S_k^- = \{j : j \in W_k\ and\ x_j = 0\}$$

$$S_k^0 = \{j : j \notin W_k\}$$

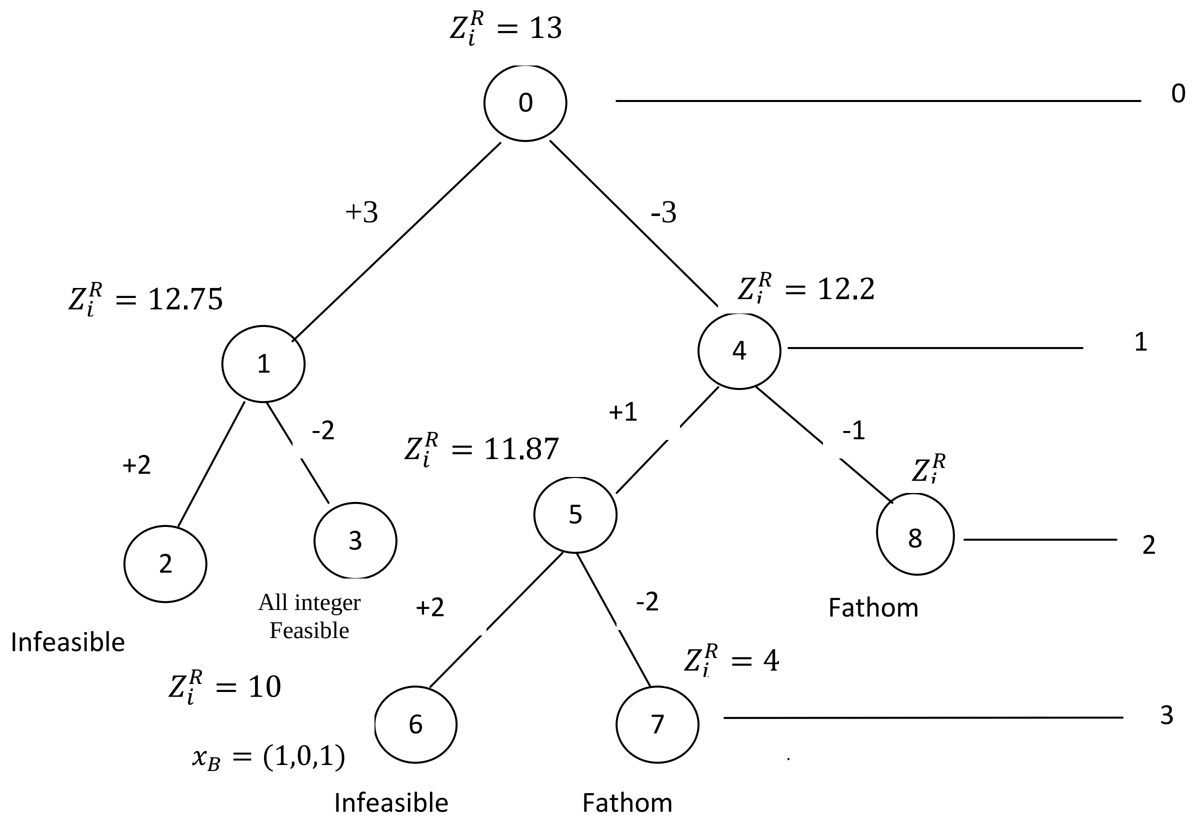

Figure 1. Search tree for knapsack problem

A completion of W_k is an assignment of binary variables to the free variables specified by the index set S_k^0 . The B&B process is guided by the following rules:

1. If a free variable remains at node k ($S_k^0 \neq \emptyset$) choose a separation variable ($j \in S_k^0$) and branch to the next level to create node k+1.
2. If no free variables remain ($S_k^0 = \emptyset$) , evaluate the solution and backtrack to the highest level that contains a node whose right branch has not been explored; generate the node along this branch one level down.

Depth-first Search:

To implement this process we need a data structure that gives the status of the tree at any point. The vector P_k will be used for this purpose. For node k at level l of the tree, P_k is defined as follows:

1. The length of the vector is l, and is written $P_k = (j_1, j_2, \ldots\ldots, j_l)$.
2. The component j_i is the separation variable at level i.
3. The sign of j_i indicates the value of the separation variable on the current path. A negative sign indicates that the variable is set to zero, and a positive sign (or no sign) indicates it is set to 1.
4. The component j_i can be underlined or not. If it is underlined, the alternative node at level i has already been explored. If it is not underlined, the alternative has yet to be explored.
5. The variables not mentioned in P_k are the free variables.

Backtracking: Underline the rightmost non underlined entry in P_k, change its sign, and erase all entries to its right. If all entries are underlined the enumeration is complete and the incumbent is the solution.

3.2 Generic B&B Algorithm for 0-1 IP

In the following algorithm, k is used as a counter for identifying nodes, l is the level in the tree, and n is the number of integer variables. The l-dimensional vector, P_k, identifies the path from the root to node k, and $S_k^+,\ S_k^-, and\ S_k^0$ indicate the current status of the variables. Vectors $x^k\ and\ x_B$ refer to a feasible solution at node k and the incumbent, respectively. (Wolsey 1998).

Initialization: Create node 0 at level 0. Let $k = 0, l = 0\ and\ P_o = \emptyset$.

Perform the approximate procedure in an attempt to find a feasible solution. If a feasible solution is found, let x_B be the vector of integer variables with objective value z_B otherwise, let $z_B = -M$

Update: If $l = n$, an integer solution has been generated. Let x^k be the the solution vector defined by P_k , if x^k is feasible, compare the objective value z^k with z_B . If $z^k > z_B, put\ x_B \leftarrow x^k\ and\ z_B \leftarrow z^k$

whether or not x^k is feasible go to *Backtrack* .If $l < n$, continue with *Variable Fixing.*

Variable Fixing: Use logical and other tests to determine if a free variable x_j should be set to 1 or 0, where $j \in S_k^0$. Suppose variable x_s should be set to 1. Perform the following:

$$l \leftarrow l + 1, k \leftarrow k + 1.$$

Add +s to the current path vector to get P_k. Create node k with decision +s.

Alternatively, suppose variable x_s should be set to 0. Perform the following

$l \leftarrow l + 1, k \leftarrow k + 1$

Add –s to the current path vector to get P_k. Create node k with decision –s.

If any changes are made repeat this step; otherwise, continue with *Bound.*

Bound: Solved the relaxed problem at node k. If the result shows no feasible solution, fathom the node and go *backtrack.*

If the procedure returns $z_{UB}^k,\ put\ z_{UB}^k \leftarrow \lfloor z_{UB}^k \rfloor$,and compare this value to the incumbent.

If $z_{UB}^k \leq z_B$, fathom the node and go to *Backtrack*; otherwise continue with *Approximate*.

Approximate: Attempt to find a feasible solution in the set of solutions for node k. If a feasible solution is found, call it x^k and let the objective value be z^k .

If $z^k \geq z_B$, put $x_B \leftarrow x_k\ and\ z_B \leftarrow z^k$.

If $z_{UB}^k = z_B$, fathom the node and go to *Backtrack*; otherwise, continue with *Branch*.

Branch: Choose a separation variable x_s, such that $s \in S_k^0$, and a direction of exploration.

Put $k \leftarrow k + 1, l \leftarrow l + 1$ and create node k at level of the tree.

If x_s is to be set to 1, add +s to the current path vector to get P_k.

If x_s is to be set to 0, add –s to the current path vector to get P_k.

Go to *Update.*

Backtrack: In the vector P_k , find the element furthest to the right that is not underlined. If all elements are underlined, stop, x_B is the optimum.

Otherwise, let this element be j_i. Backtrack by doing the following:

Let the level be $l = i$.

Delete all elements of P_k to the right of j_i.

Put $j_i \leftarrow -j_i$ (i.e change the sign of j_i).

Now underline j_i to get $P_{k+1.}$

Put $k \leftarrow k + 1$ and create node k.

Go to Update.

Table 2 below shows the results of the algorithm when applied to the knapsack example given in (Fig.1), we assume that no variable fixing or Approximate procedures were used. During the bound step, the LP relaxation was solved using the "bang for buck" scheme. At each node, the free variable with the largest benefit/cost ratio was chosen for separation. And always set equal to one.

Table 2. B&B results for knapsack example

Node,k	Level,l	P_k	z_{UB}	z_B	x_B	s	Action
0	0	$\emptyset$	13	-∞	-	3	Set $x_3 = 1$
1	1	(+3)	12	-∞	-	2	Set $x_2 = 1$
2	2	(+3,+2)	Infeasible	-∞	-	-	Backtrack
3	2	(+3,$-\underline{2}$)	10	10	(1,0,1)	-	Feasible Backtrack
4	1	($-\underline{3}$)	12	10	(1,0,1)	1	Set $x_1 = 1$
5	2	($-\underline{3}$,+1)	11	10	(1,0,1)	2	Set $x_2 = 1$
6	3	($-\underline{3}$,+1,+2)	Infeasible	10	(1,0,1)	-	Backtrack
7	3	($-\underline{3}$,+1, $-\underline{2}$)	4	10	(1,0,1)	-	Fathom Backtrack
8	2	($-\underline{3}$, $-\underline{1}$)	9	10	(1,0,1)	-	Fathom Backtrack Finish

Table 2 also gives summary of the computations involve in the algorithm; for instance, we obtain node 4, by putting ($k \leftarrow k + 1, l \leftarrow l + 1$) at that level, where $k \leftarrow 3 + 1, l \leftarrow 0 + 1$, (i.e $k \leftarrow 4, l \leftarrow 1$). We obtain $P_k = -\underline{3}$; since $x_s = x_2$ $is\ been\ fixed\ to\ zero$, therefore we add -2 to the current path vector to get P_k; with objective function value $z_B = 10$ $and\ integer\ variables, x_B = (1,0,1)$.

Conclusion

This paper gives an over view and demonstrate the problem solving technique of one of the seemingly trivial or simple problem, but actually very tricky and difficult to solve; the knapsack problem; defined as a Binary Integer Programming problem. The Generic algorithm presented will go a long way in helping researchers and students to generalize the B&B approach taken for 0-1 IPs when depth-first search is used; especially for a general branching with integer variables other than 0-1.

References

[1] Abdullahi,.N.(2006),"Adopting an MIP model in military training: a case study of the US Army "Training Mix Model" in the Nigerian Defence Academy, Kaduna" MSc Thesis; Mathematics Department, Ahmadu Bello University, Zaria.

[2] Achterberg,T., Koch,T. and Martin,A.(2004) "Branching Rules Revisited" Zuse Institute Berlin,(ZIB)-Report 04-13.

[3] Balas,E.(1965) "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", *Operations Research*, **13**,.***(4)***, 517-546.

[4] Dakin, R.J.(1965) "A Tree Search Algorithm for MIP Problems", *Computer journal*, **8**, ***(3)***, 250-255.

[5] Geoffrion, A.M.(1969)"An Improved Implicit Enumeration Approach for IP" *Operations Research*",**17**,***(3)***,437-454.

[6] Glover,F.(1965) "A multi Phase-Dual Algorithm for the Zero-One IPP", *Operations Research*,**13**, ***(6)***, 879-919.

[7] Glover,F.(1968) “Surrogate Constraints” *Operations Research*, **16**, ***(4)***,741-749.

[8] Land, A.H. and Doig, A.G.(1960) “An automatic method for solving Discrete Programming Problems”, *Econometrica*, **28**, ***(3),*** 497-520.

[9] Lee,E.K[1]. and Mitchel,J.E[2],(1998) “Branch and Bound Methods for Integer Programming” Notes: [1]Industrial and system Engineering, Georgia Institute of Technology, Atlanta, USA; [2]Mathematical Sciences, Rensselaer polytechnic Institute, Troy NY USA. AMS 1991 Subject classification:90C10 ,49M35, 90C11, 90C06.

[10] Mitten,L.G.(1970) “Branch and Bound Methods: General Formulation and Properties”, *Operations Research*, **18**, ***(1)***, 24-34.

[11] Mitchel,J.E,(1995) “Interior Point Methods for Combinatorial Optimization”, RPI Mathematical Sciences Report No.217.

[12] Nauss, R.M. (1976) “An Efficient Algorithm for the 0-1 Knapsack Problem”, *Management Science,* **23**, ***(1)***,1-15.

[13] Nemhauser,G.L. and Wolsey,L.A.(1988) “Integer and Combinatorial Optimization,Wiley, New York-US.

[14] Schrijver,A.(1986) ”Theory of Linear and Integer Programming” John Wiley, Chichester.

[15] Wolsey,L.A(1998) ”Integer Programming”, John Wiley, New York.