GANIT *J. Bangladesh Math. Soc.* (ISSN 1606-3694) Vol. 33 (2013) 1 – 11

A DECOMPOSITION TECHNIQUE FOR SOLVING INTEGER PROGRAMMING PROBLEMS

Md. Istiaq Hossain[1], and M. Babul Hasan[2]
[1] *Institute of Natural Science, United International University, Dhaka, Bangladesh*
[2] *Department of Mathematics, University of Dhaka, Bangladesh*
E-mail: [1]shishir2004x@gmail.com , [1] mbabulhasan@yahoo.com

Received 09-05-2012 Accepted 31-07-2013

ABSTRACT

Dantzig-Wolfe decomposition as applied to an integer program is a specific form of problem reformulation that aims at providing a tighter linear programming relaxation bound due to the non-convexity of an integer problem. In this paper, we develop an algorithm for solving large scale integer program relying on column generation method. We implemented our algorithm for solving Capital budgeting and scheduling type problems. Moreover, we used the Computer Aided System (CAS) AMPL to convert our algorithm into programming codes and illustrated the same problem in our program. We demonstrate our method by illustrating some numerical examples.

Keywords: Decomposition, Relaxation, Integer linear programming, Binary integer programming

1. Introduction

In the linear-programming (LP) models the variables possess *continuous* values, in the sense that decision variables are allowed to be fractional. However, when fractional solutions are not realistic, and we must consider the optimization problem as:

$$\left.\begin{aligned} &\text{Maximize } \sum_{j=1}^{n} c_j x_j \\ &\text{subject to } \sum_{j=1}^{n} a_{ij} x_j \le b_i \ (i = 1,2,\dots\dots,m) \\ &x_j \ge 0 \text{ and integer } (j = 1,2,\dots\dots,n) \end{aligned}\right\} \tag{1.1}$$

This problem is called the *integer- linear -programming problem* (ILP). It is said to be a *mixed* integer program (MIP) when some, but not all, variables are restricted to be integer, is called a *pure* integer program when all decision variables must be integers and is called a *binary* (or 0-1) integer programming (BIP) when the decision variables are either zero (0) or one (1).

Although several algorithms have been developed for ILP, none of these methods are totally reliable from the computational standpoint, particularly as the number of integer variables increases. Unlike LP, where problems with thousands of variables and thousands of constraints can be solved in a reasonable amount of time, computational experience with ILP, after more than 30 years of development, remains elusive. The computational difficulty with available ILP algorithms has led users to find other means to "solve" the problem. One such approach is to solve the model as a continuous LP and then round the optimum solution to the closest feasible integer values. However, it is nonsensical to deal with fractional values of x where the decision to finance a project can be represented by the binary variable $x = 0$ if the project is rejected and $x = 1$ if the project is accepted. The use of rounding as an approximation is logically unacceptable.

Dantzig-Wolfe decomposition [2, 3] transforms the original mathematical problem into a master problem, where the number of columns may be large but the number of rows is reduced. To make the new model more tractable, columns are generated iteratively in the hopes of only having to include a subset of the columns in the model. This is denoted by delayed column generation and consists of solving a pricing problem in each iteration. Dantzig-Wolfe decomposition (DWD) as applied to an IP is a specific form of problem reformulation that aims at providing a tighter LP relaxation bound. The reformulation gives rise to an integer master problem, whose typically large number of variables is dealt with implicitly by using an IP column generation procedure, also known as branch-and-price algorithm.

Column generation (CG) can be considered as one of the most successful approaches for solving large-scale IP problems over the last decade. The idea was introduced by Gilmore & Gomory [8] when finding the optimal solution for the cutting-stock problem. The problem was to cut lumber which were in different lengths to meet demands for specific lengths of shelves. They noticed that, it is practically impossible to generate all the possible configurations of lumber cuts. Therefore, they devised a methodology which started with a set of feasible configurations, and used the optimal solution to the simplified problem to generate a new promising configuration. Later, the idea has been used by Minoux [11] as a powerful technique to reformulate some important combinatorial problems. Vance et. al. [15] combined CG and branch-and-bound to present an algorithm for binary cutting stock problems. In this article, we have studied to solve the BIP in an efficient and user friendly way. For this, we have used the concept of DWD principle relying on column generation along with Lagrangean relaxation for solving the problem.

The solution method of LP problems by using decomposition are discussed in Hossain [9] based on the notion of searching for optimal solution to a problem among the near-optimal solutions to its Lagrangean relaxation. The decomposition method for IP is discussed effectively in Sweeny [13] which presents the method for decomposing large scale IP (more than 200 variables) having a block angular structures. For the IP problem, the consequence of the non-convexity is that we may not have an optimal solution for the original problem. Application of Lagrangean relaxation to a block angular integer problem leads to smaller sub problems. A master problem is then constructed containing one column for each of the near-optimal sub problem solutions. Since for IP, we are

guaranteed only weak duality, we may not have an optimal feasible solution but we can have a good bound for every IP.

The rest of the article is organized as follows. In Section 2, we present some of the preliminary definitions. Section 3 and 4 are dedicated for some common applications of BIP models. In Section 5, 6, 7, we discuss our proposed algorithm, working steps and flowchart of the solution method. Numerical problems of some BIP models are illustrated in section 8. The pseudo codes in AMPL [6] for the method and the outputs are shown in Section 9. In Section 10, we discussed our results and findings. Finally, we draw a conclusion.

2. Some Preliminary Definitions

In this section, we will discuss some elementary definitions and its mathematical expression to pursue our problem of interest.

2.1 Lagrangean relaxation

To describe the Lagrangean relaxation [5, 7], let us consider the general BIP as

$P(x)$: Maximize cx, $Ax \leq b$, $Dx \leq f$, $x\epsilon\{1,0\}$.

Then the Lagrangean relaxation, for fixed λ is defined as

$S(x,\lambda)$: Maximize $cx - \lambda(Ax - b)$, $Dx \leq f$, $x\epsilon\{1,0\}$.

For minimization problem let the problem is given as follows

$P(x)$: Minimize cx, $Ax \geq b$, $Dx \geq f$, $x\epsilon\{1,0\}$.

Then the Lagrangean relaxation, for fixed λ is defined as

$S(x,\lambda)$: Maximize $cx + \lambda(Ax - b)$, $Dx \geq f$, $x\epsilon\{1,0\}$.

2.2 Weak vs. Strong Duality

Consider the IP problem given by the system (1.1). Then the Lagrangean relaxation $S(x,\lambda)$ of P is defined as $S(x,\lambda)$: minimize $cx + \lambda(b - Ax), Dx = f, x \geq 0$.

In a particular case if x is feasible in P, then $Ax - b = 0$, so $cx + \lambda(b - Ax) = cx$.

So everything feasible to P has the same objective value in S. But S has a larger feasible region. So $v(S) \leq v(P)$. This is weak duality.

On the other hand for LPs, $v(S) = v(P)$ at the optimum. This is strong duality.

3. Application for Binary Integer Programming (BIP)

In the last decade, the column generation approach to IP has been successfully applied to several problems that can be formulated as some variant of the set partitioning problem with binary variables. These applications include routing problems Desrosiers et al. [4], crew assignment problems Anbil et al. [1], the generalized assignment problem Savelsberg [12], edge clustering problems Johnson et al. [10], and Vanderbeck [17], and the bin packing problem Vance et al. [15] and Vanderbeck [18] among others. In the present paper, as well as in the recent studies of Vance [16] and Valerio de Carvalho [14], cutting stock problems are solved to optimality using column generation within a branch-and-bound algorithm. Such applications, where the master obtained through Dantzig-

Wolfe decomposition is an integer program with non-binary variables, emphasize the difficulties inherent to branching in an IP column generation approach.

4. Some Binary Integer Programming (BIP) Models

Integer-programming models arise in practically every area of application of mathematical programming. To develop a preliminary appreciation for the importance of these models, we introduce, in this section, two areas namely Capital Budgeting and Scheduling problems where the BIP has played an important role in supporting managerial decisions. We do not provide the most intricate available formulations in each case, but rather give basic models and suggest possible extensions.

Capital Budgeting

In a typical capital-budgeting problem, decisions involve the selection of a number of potential investments. The investment decisions might be to choose among possible plant locations, to select a configuration of capital equipment, or to settle upon a set of research and development projects. Often it makes no sense to consider partial investments in these activities, and so the problem becomes a *go–no-go* integer program, where the decision variables are taken to be $x_j = 0$ or 1, indicating that the j-th investment is rejected or accepted. Assuming that c_j is the contribution resulting from the j-th investment and that a_{ij} is the amount of resource i , such as cash or manpower, used on the j-th investment, we can state the problem formally as:

$$\left.\begin{aligned} &\text{Maximize } \sum_{j=1}^{n} c_j\, x_j \\ &\text{subject to } \sum_{j=1}^{n} a_{ij}\, x_j \le b_i \ (i = 1,2,\dots\dots,m) \\ &x_j = 0 \text{ or } 1 \ (j = 1,2,\dots\dots,n) \end{aligned}\right\} \qquad (4.1)$$

The objective is to maximize total contribution from all investments without exceeding the limited availability b_i of any resource.

The simplest of all capital-budgeting models has just one resource constraint, but has attracted much attention in the management-science literature. It is stated as:

$$\text{Maximize } \sum_{j=1}^{n} c_j\, x_j,$$

subject to:

$$\sum_{j=1}^{n} a_{ij}\, x_j \le b_i \qquad (i = 1,2,\dots\dots,m),$$

$$x_j = 0 \text{ or } 1 \qquad (j = 1,2,\dots\dots,n)$$

Usually, this problem is called the 0–1 *knapsack* problem, since it is analogous to a situation in which a hiker must decide which goods to include on his trip. Here $c\,j$ is the ''value'' or utility of including good j, which weighs $a_j > 0$ pounds; the objective is to maximize the ''pleasure of the trip,'' subject to the weight limitation that the hiker can carry no more than b pounds.

Scheduling

The entire class of problems referred to as sequencing, scheduling, and routing is inherently integer programs. Consider, for example, the scheduling of students, faculty, and classrooms in such a way that the number of students who cannot take their first choice of classes is minimized. There are constraints on the number and size of classrooms available at any one time, the availability of faculty members at particular times, and the preferences of the students for particular schedules. Clearly, then, the i-th student *is* scheduled for the j-th class during the n-th time period or not; hence, such a variable is either zero or one.

As a specific example, consider the scheduling of airline flight personnel. The airline has a number of routing ''legs'' to be flown, such as 10 A.M New York to Chicago, or 6 P.M. Chicago to Los Angeles. The airline must schedule its personnel crews on routes to cover these flights. One crew, for example, might be scheduled to fly a route containing the two legs just mentioned. The decision variables, then, specify the scheduling of the crews to routes:

$$x_j = \begin{cases} 1 & \text{if a crew is assigned to route j} \\ 0 & \text{otherwise.} \end{cases}$$

Let

$$a_{ij} = \begin{cases} 1 & \text{if leg i is included on route j} \\ 0 & \text{otherwise.} \end{cases}$$

And

$$c_j = \text{Cost for assigning a crew to route j.}$$

The coefficients a_{ij} define the acceptable combinations of legs and routes, taking into account such characteristics as sequencing of legs for making connections between flights and for including in the routes ground time for maintenance. The model becomes:

$$\text{Maximize} \sum_{j=1}^{n} c_j\, x_j,$$

subject to:

$$\sum_{j=1}^{n} a_{ij}\, x_j = 1 \qquad (i = 1,2,\ldots\ldots,m),$$

$$x_j = 0 \text{ or } 1 \qquad (j = 1,2,\ldots\ldots,n).$$

The i-th constraint requires that one crew must be assigned on a route to fly leg i. An alternative formulation permits a crew to ride as passengers on a leg.

5. Algorithm for Solving Binary Integer Programming

Let the original integer problem is stated as system (4.1), where the variables are assumed to have the values either 0 or 1 (i.e. binary). Then we have the following simple steps as our algorithm.

Initialize Set iteration $k = 1$ and pick a λ^k

Step: 1 Solve sub-problem $s(x, \lambda^k)$: max $cx - \lambda^k(Ax - b)$, $Dx \leq f, x \in \{0,1\}$

These variables are x.

Step: 2 Take the solution x^k and put in $M(\theta)$ with a new variable θ_k

Solve $M(\theta)$: max $\sum_{l=1}^{k} \theta_l \, cx^l$,

$\sum_{l=1}^{k} \theta_l \, a_l x^l \leq b_i$ for each i ,

$\sum_{l=1}^{k} \theta_l = 1$,

$\theta_l \geq 0$.

We get solution θ^* and dual prices λ^*.

Step: 3 If $v\left(S\left(x, \lambda^k\right)\right) = v(M(\theta)$, then stop.

Else set $k = k + 1$, set $\lambda^k = \lambda^*$, and go to Step 1.

6. Working Steps

The followings are the working steps for getting a feasible upper bound of the given integer problem:

(i) At the very first time we have to randomly pick up an initial value of the dual variable and then solve the sub problem with respect to the variable $x_i's$ using the step-1 of the algorithm.

(ii) After we get the solution, we will import our current solution of the sub problem to form the master problem with respect to the variable $\theta_i's$ with the help of step-2 of the algorithm and solve.

(iii) Then we have to investigate the optimality condition. If the optimality condition holds, we have the feasible solution from the final sub problem values and put this in the original objective function to get the bound (upper bound for maximization and lower bound for minimization type problems). Otherwise, we will take the current dual value from the master problem and import this to update our sub problem and continue the same process unless we meet the optimality condition.

6.1 Flowchart

The following is a pictorial representation of the communication between master problem (restricted master) and sub-problems discussed earlier.

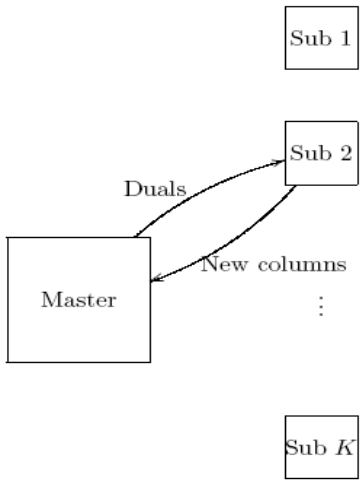

FIGURE 1. Communication between restricted master and sub-problems

7. Numerical examples

In this section, we will give two numerical examples in which the exact solution is shown as a remark and the complete solution procedure is shown for the first example only.

(A)

$$\begin{aligned} \text{Maximize } z &= 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5 \\ \text{Subject } to \quad 5x_1 &+ 4x_2 + 3x_3 + 7x_4 + 8x_5 \le 25 \\ x_1 &+ 7x_2 + 9x_3 + 4x_4 + 6x_5 \le 25 \\ 8x_1 &+ 10x_2 + 2x_3 + x_4 + 10x_5 \le 25 \\ x_i &\ge 0 \end{aligned} \tag{7.1}$$

Remark 1:

The exact solution of the problem is $(x_1, x_2, x_3, x_4, x_5) \equiv (1,1,1,1,0)$ and the maximum of z is$= 95$.

Solution:

Applying the Lagrangean relaxation by relaxing constraint (7.1), we have the general sub-problem as follows:

Sub-problem for $k-th$ iteration

Maximize $z = 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5 - \lambda_k(5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 - 25)$

Subject to $x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 \le 25$; $8x_1 + 10x_2 + 2x_3 + x_4 + 10x_5 \le 25$; $x_i \in \{0,1\}$.

Master problem for $k-th$ iteration

Maximize $z = \sum_{i=1}^{k} \theta_i \, (20x_1^k + 40x_2^k + 20x_3^k + 15x_4^k + 30x_5^k)$

Subject to $\sum_{i=1}^{k} \theta_i \left(5x_1^k + 4x_2^k + 3x_3^k + 7x_4^k + 8x_5^k\right) \le 25$

$$\sum_{i=1}^{k} \theta_i \left(x_1^k + 7x_2^k + 9x_3^k + 4x_4^k + 6x_5^k\right) \leq 25$$
$$\sum_{i=1}^{k} \theta_i \left(8x_1^k + 10x_2^k + 2x_3^k + x_4^k + 10x_5^k\right) \leq 25$$
$$\sum_{i=1}^{k} \theta_i = 1;\ \theta_i \geq 0$$

Now we are about to start the iterating process.

Iteration-1 (For k=1)

Starting value for $\lambda_1 = 100$

Sub-problem

Maximize $z = 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5 - \lambda_k(5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 - 25)$

$$= -80x_1 - 360x_2 - 280x_3 - 685x_4 - 770x_5 + 2500$$

Subject to $x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 \leq 25$; $8x_1 + 10x_2 + 2x_3 + x_4 + 10x_5 \leq 25$;$x_i \in \{0,1\}$

Solving by Lindo [19] gives all $x_i's$ are zero and sub-problem value max $z = S_1(v) = 2500$.

Master problem

Maximize $z = \theta_1(20.0 + 40.0 + 20.0 + 15.0 + 30.0) = 0.\theta_1$

Subject to $\theta_1(5.0 + 4.0 + 3.0 + 7.0 + 8.0) = 0.\theta_1 \leq 25$; $\theta_1 = 1$; $\theta_i \geq 0$

Solving by Lindo gives $\theta_1 = 1$, and master problem value max $z = M_1(v) = 0$, and the dual value for the next step is $\lambda_2 = 0$

Since $z = S_1(v) = 2500 \neq 0 = M_1(v)$, thus the current solution is not optimal. Hence we proceed to the next iteration.

Iteration-2 (For k=2)

Sub-problem

Maximize $z = 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5 - 0(5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 - 25)$

$$= 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5$$

Subject to $x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 \leq 25$

$$8x_1 + 10x_2 + 2x_3 + x_4 + 10x_5 \leq 25;\ x_1, x_2, x_3, x_4 \in \{0,1\}.$$

Solving by Lindo gives $x_1 = x_2 = x_3 = x_4 = 1$, $x_5 = 0$ and the sub-problem value $z = S_2(v) = 95$.

Master problem

Maximize $z = 0.\theta_1 + 95\theta_2 = 95\theta_2$

Subject to $0.\theta_{1+} + 19\theta_2 \leq 25$; $0.\theta_1 + 21\theta_2 \leq 25$; $0.\theta_1 + 22\theta_2 \leq 25$;

$$\theta_1 + \theta_2 = 1;\ \theta_1, \theta_i \geq 0$$

Solving by Lindo gives $\theta_1 = 0, \theta_2 = 1$ and master problem value $z = M_2(v) = 95$.

Since $z = S_2(v) = 95 = 95 = M_2(v)$, thus the current solution is optimal. This gives $x_1 = x_2 = x_3 = x_4 = 1,\ x_5 = 0$ with maximum of $z = 95$.

B. Covering All Characteristics (Assignment/Scheduling problem):

Southwestern Airways needs to assign its crews to cover all its upcoming flights. We will focus on the problem of assigning crews based in San Francisco to the flights listed in the first column of the Table 1. The other 12 columns show the 12 feasible sequences of flights for a crew. We have to choose the crew in such a way that every flight is covered. The cost of assigning crew to a particular sequence of flights is given (in thousands of dollars) in the bottom row of the table. The objective is to minimize the total cost that covers all the flights.

Flight	Feasible Sequence of Flights											
	1	2	3	4	5	6	7	8	9	10	11	12
1. San Francisco to Los Angeles	1			1			1			1		1
2. San Francisco to Denver				1			1				1	
3. San Francisco to Seattle			1			1				1		1
4. Los Angeles to Chicago				2		2			3	2		3
5. Los Angeles to San Francisco	2					3					5	5
6. Chicago to Denver				3	3					4		
7. Chicago to Seattle						3	3			3	3	4
8. Denver to San Francisco		2		4	4					5		
9. Denver to Chicago			2					2	2			
10. Seattle to San Francisco			2				4	4				5
11. Seattle to Los Angeles					2				2	4	4	2
Cost $1,000's	2	3	4	6	7	5	7	8	9	9	8	9

Table 1: Data for Southwestern Airways Problem

Formulation with Binary variables:

With 12 feasible sequences of flights, we have 12 yes-or-no decisions. For example, consider the last flight in the Table 1 (Seattle to Los Angeles). Five sequences (namely, sequences 6, 9, 10, 11 and 12) include the flight. Therefore, at least one of these five sequences must be chosen. The resulting constraint is

$$x_6 + x_9 + x_{10} + x_{11} + x_{12} \geq 1$$

Using similar constraints for the other 10 flights, the complete BIP model is

Minimize $Z = 2x_1 + 3x_2 + 4x_3 + 6x_4 + 7x_5 + 5x_6 + 7x_7 + 8x_8 + 9x_9 + 9x_{10} + 8x_{11} + 9x_{12}$

Subject to

$$x_1 + x_4 + x_7 + x_{10} \geq 1; \qquad x_2 + x_5 + x_8 + x_{11} \geq 1$$

$$x_3 + x_6 + x_9 + x_{12} \geq 1; \qquad x_4 + x_7 + x_9 + x_{10} + x_{12} \geq 1$$
$$x_1 + x_6 + x_{10} + x_{11} \geq 1; \qquad x_4 + x_5 + x_9 \geq 1$$
$$x_7 + x_8 + x_{10} + x_{11} + x_{12} \geq 1; \; x_2 + x_4 + x_5 + x_9 \geq 1$$
$$x_5 + x_8 + x_{11} \geq 1; \qquad x_3 + x_7 + x_8 + x_{12} \geq 1$$
$$x_6 + x_9 + x_{10} + x_{11} + x_{12} \geq 1$$

and x_j is binary, for $i = 1,2, \ldots \ldots, 12$.

Remark 2:

The exact solution of the problem is $x_3 = x_4 = x_{11} = 1$ and other $x_i's$ are zero. Another optimal solution is $x_1 = x_5 = x_{12} = 1$ other $x_i's$ are zero. The minimum of z is= 18,000$.

8. Pseudo Codes

We develop a pseudo code based in AMPL [6]. Every AMPL program is consists of three parts namely model, run and data file. The readers are referred to the authors for the program files.

9. Findings & Result Discussions

The result shown as remark 1 and remark 2 in section 7(A) and 7(B) respectively was found directly by using the optimization software Lindo. If we closely take a look at the results in those sections and that we have computed using our AMPL codes, we can find that algorithm is giving precisely the same result in example (A) and giving a good lower bound for the second one. For the first problem, we had 5 variables and three constraints which converged to the exact solution in only two iterations. But for the second problem, we had relatively large scale problem with 12 variables and 11 constraints which is actually a quite difficult task to solve exactly using the conventional method of solving binary integer programming. Though our method does not offer exact solution, it can be incredibly useful to study the feasibility of a large scale BIP. Moreover the algorithm has an available pseudo code to turn into a computer program which can help our searching faster.

10. Conclusion

Our decomposition technique has a noteworthy advantage that may or may not provide an optimal solution but it gives us a tighter bound for solving large scale integer programs. We implemented our algorithm for solving Capital budgeting and scheduling type problems. Furthermore, our pseudo codes facilitate our algorithm to convert into an AMPL codes for getting computer generated solutions. We showed the numerical problems manually and also using our codes which make our method more interactive.

REFERENCES

[1] Anbil, R., C. Barnhart, L. Hatay, E. L. Johnson, V. S. Ramakrishnan (1993), *"Crew-pairing optimization at American Airlines Decision Technologies"*, T.A. Cirani, R.C. Leachman, eds. Optimization in Industry. Wiley, New York, pp. 31–36.

[2] Dantzig, G, B (1993), "*Linear Programming and Extensions*", Princeton University Press, Princeton, U.S.A.

[3] Dantzig, G,B and P. Wolfe (1996), "*The Decomposition Algorithm for Linear Programming*", Econometrica, Vol. 29, No. 4.

[4] Desrosiers, J., Y. Dumas, M. M. Solomon, F. Soumis. (1994), Time constrained routing and scheduling. M.E. Ball, T.L. Magnanti, C. Monma, G.L. Nemhauser, eds. Handbooks in Operations Research and Management Sciences: Networks. North-Holland, Amsterdam.

[5] Fisher, M.L (1979), "*The Lagrangean Relaxation Method for Solving Integer Programming Problems*", Management Science, Vol. 27, No. 1.

[6] Fourer, R., D.M. Gay and B.W. Kernighan (2003), A Modeling Language for Mathematical Programming, Second Edition, Thomson Publication.

[7] Geofrion, A.M (1974), "*Lagrange Relaxation for Integer Program*", Mathematical Programming Study, Vol. 2, pp. 82-114.

[8] Gilmore, P. C., and Gomory, R. E., 1961, *"A linear programming approach to the cutting stock problem"*, Operations Research 9, pp. 849-859.

[9] Hossain, M. I.and Hasan, M. B. (2011), *"An Improved Decomposition Algorithm and Computer Technique for Solving LPs*", IJBAS-IJENS, Vol.11, Issue. 3, pp. 14-25.

[10] Johnson, E. L., A. Mehrotra, G. L. Nemhauser (1993), "*Min-cut clustering*", Math. Programming 62, pp. 133–151.

[11] Minoux, M. 1987, A class of combinatorial optimization problems with polynomially solvable large problem, Operations Research, Vol. 9, pp. 849–859.

[12] Savelsbergh, M. W. P., (1997), *"A branch-and-price algorithm for the generalized assignment problem*", Operations Research 45, pp. 831–84.

[13] Sweeny, D.J. and Murphy (1979), "*A Method of Decomposition for Integer Programs*", Operations Research, Vol. 27, No. 6, pp. 1128-1141.

[14] Valerio de Carvalho, J. M. (1996), "*Exact solution of bin-packing problems using column generation and branch-and-bound*", Working Paper, Depart. Prouducao e Sistemas, Universidade do Minho, Portugal.

[15] Vance, P.H., C. Barnhard, E.L. Johnson, G.L. Namhauser (1994), "*Solving Binary Cutting Stock Problems by Column Generation and Branch-and-Bound*", Comput. Optim. Appl. 3, pp. 111-130.

[16] Vance, P.H., C. Barnhard, E.L. Johnson, G.L. Namhauser (1996), "*Branch-and-price algorithms for the one-dimensional cutting stock problem*", Working Paper, Department of Industrial Engineering, Auburn University, Auburn, AL. To appear in Comput. Optim. Appl.

[17] Vanderbeck, F. (1994), "*Decomposition and column generation for integer programs*", Ph.D. Thesis. Faculte des Sciences Appliqees, Universite Catholique de Louvain, Louvain-la-Neuve.

[18] Vanderbeck, F. (1996), "*Computational study of a column generation algorithm for bin packing and cutting stock problems*", Research Papers in Management Studies, University of Cambridge, to appear in Math. Programming.

[19] www.lindo.com