An Improved Decomposition Technique for Solving Integer Linear Programming Problems

M. A. I. Bhuiyan
Department of Mathematics, University of Barisal
Barisal, Bangladesh

M. S. Hossain
Department of Mathematics, University of Barisal
Barisal, Bangladesh

ABSTRACT
Decomposition technique is one of the most frequently used technique for solving Linear Programming Problems (LPPs) as well as Integer Linear Programming Problems (ILPPs). There are many existing techniques for solving ILPPs such as Branch and Bound method, Cutting Plane method etc. The purpose of this paper is to develop computer oriented decomposition technique for solving ILPPs using benders decomposition method. Applying decomposition technique anyone can solve ILPPs by dividing original problem into two easier problems, namely Master problem and Sub problem. This paper proposes a new technique for solving a ILLP manually and develops a computer code using a Mathematical Programming Language (AMPL). Also a comparison of manual output and programming output has been presented.

Keywords
LPP, ILPP, DWD, Benders Decomposition, AMPL, Master-Problem, Sub-Problem.

1. INTRODUCTION
The development of linear programming (LP) is the most scientific advances in the mid 20th century. LP grips the planning of activities to obtain an optimal result which reaches the specialized goal "*the best among all feasible alternatives*". Numerous algorithms for solving LP problem have been developed in the past. The Integer Linear Programming Problem (ILPP) is one of the latest LP problems which refer to the class of combinatorial constrained optimization problems with integer variables, where the objective function is a linear function and the constraints are linear inequalities.

A wide variety of real life problems in logistics, economics, social science and politics can be formulated as linear integer optimization problems. The combinatorial problems, like the knapsack-capital budgeting problem, warehouse location problem, travelling salesman problem, decreasing costs and machinery selection problem, network and graph problems, such as maximum flow problems, set covering problems, matching problems, weighted matching problems, spanning trees problems and many scheduling problems can also be solved as linear integer optimization problems.

Benders Decomposition is one of the popular techniques for solving certain classes of difficult problems such as stochastic programming problems and mixed-integer linear programming problems. Benders Decomposition is a technique in mathematical programming that allows the solution of very large linear programming problems that have a special block structure. This structure often occurs in applications such as stochastic programming. As it process towards a solution, Benders decomposition adds new Constraints , So the approach is called "row generation" .In contrast, Dantzig-Wolfe Decomposition (DWD) uses "Column generation".

In 2011, Md. Istiaq Hossain and Md. Babul Hasan [1] developed an improved decomposition algorithm for solving large scale LPs depending on DWD principle. Also, in 2013 H. K Das and Hasan [2] developed a primal dual approach of linear fractional Programming (LFP) and LP problem depending on DWD principle. But they did not mention the behavior of their algorithm in case of ILPPs.

The outline of this paper is like in Section 2, some basic ideas and necessary definitions related to the work have been mentioned. In Section 3, some existing techniques to solve ILPPs have been presented. In section 4, an improved algorithm has developed to solve ILPPs based on Benders Decomposition. In Section 5, 6 & 7, a ILPP has been solved by using proposed algorithm manually and generate a computer code using AMPL. In Section 8, a tabular comparison between manual output and programming output has presented. Finally in Section 9, convergences of Master and Sub problem values have been drawn graphically [3].

2. PRELIMINARIES
In this current section some basic definitions have been discussed which are relevant to the work.

2.1 General Linear Programming Problem (LPP)
The general linear programming problem (LPP) is to find the decision variables $x_1, x_2, , , , , x_n$ which is optimizing (minimizing or maximizing) the objective function.

$z = c_1x_1 + c_2x_2 \dots\dots\dots\dots + c_nx_n$

Subject to the constraints,

$a_{11}x_1 + a_{12}x_2 \dots\dots\dots + a_{1n}x_n \ (\geq, =, \leq)\ b_1$

$a_{21}x_1 + a_{22}x_2 \dots\dots\dots + a_{2n}x_n \ (\geq, =, \leq) b_2$

……

……

$a_{m1}x_1 + a_{m2}x_2 \dots\dots\dots + a_{mn}x_n (\geq, =, \leq) b_m$

$and\ x_j \geq 0; \quad j = 1,2,3, , , , , , , n.$

The coefficients $c_j\ (j = 1,2,3, \dots\dots, n)$ are called the cost coefficients. The constants $b_i (i = 1,2,3, \dots\dots, m)$ in the constraints conditions are called stipulations and the constants $a_{ij}\ (i = 1,2,3, \dots\dots, m, j = 1,2,3, \dots\dots, n)$ are called the structural coefficients.

2.2 Integer Programming (IP)

The problem at which all or some of the variables are required to be integers is called an Integer programming problem. An IP in which all of the variables are required to be integers is called a Pure IP problem. An IP in which only some of the variables are required to be integer is called a Mixed IP problem. An IP in which all of the variables must equal to 0 or 1 is called a 0-1 IP.

2.3 Duality in Linear Programming (LP)

Every linear programming problem whether it is of maximization or minimization is associated with its mirror image problem based on the same data. The original problem is often termed as primal problem while its image problem is called as its dual problem. However, in general either problem can be considered as primal and the remaining as the dual problem. Moreover, a solution to the primal problem also gives a solution to the dual problem and vice versa. Duality is an extremely important and interesting feature of linear programming.

2.4 Benders Decomposition

Benders Decomposition is a technique in mathematical programming that allows the solution of very large linear programming problems that have a special block structure.

Benders Decomposition Principle for Linear Programming (LP)

Original Problem

$$(P) \quad \begin{aligned} &Maximize && cx + qy \\ &subject\ to && Ax \le b \\ & && Tx + Hy \le b \\ & && x, y \ge 0 \end{aligned}$$

Apply Benders decomposition principle to compose this into the following master problem and sub-problem taking the dual of sub-problem.

Master problem

$$(M) \quad \begin{aligned} &Maximize && cx + v(S(x)) \\ &subject\ to && Ax \le b \\ & && x \ge 0 \end{aligned}$$

Sub-problem

Primal Sub-problem

$$\begin{aligned} &Maximize && qy \\ &subject\ to && Hy \le b - Tx, \ \ depends\ on\ x. \\ & && y \ge 0 \end{aligned}$$

Dual Sub-problem

$$(S(x)) \quad \begin{aligned} &Minimize && \lambda(b - Tx) \\ &subject\ to && \lambda H \ge q \\ & && \lambda \ge 0 \end{aligned}$$

3. EXISTING TECHNIQUES

In this section, some existing techniques have discussed briefly for solving ILPPs.

3.1 Branch and Bound Method

Branch and bound method is applicable for pure as well as mixed integer programming problems. It has two parts. The first part is the procedure involving partitioning is called *branching* while the second part of establishing limit is referred to as bounding. The iterative procedures of branch and bound method are as follows [4]:

Step 1: Obtain the optimal solution to the given ILPP ignoring the restriction of integers.

Step 2: Test the integrality of the optimal solution obtained in step 1. There are two cases:

(a) If the solution is integers, the current solution is optimal to the given ILLP.
(b) If the solution is not integers, go to next step.

Step 3: Subdivide the given ILPP into following two sub problems considering $[x_j^*]$ is the integer part of the optimal value x_j^*.

Sub problem 1: given ILPP with an addition constraint $x_j^* \le [x_j^*]$.

Sub problem 2: given ILPP with an addition constraint $x_j^* \le [x_j^*] + 1$.

Step 4: Solve the two sub problems obtained in step 3 and check the integrality of the optimal value occurs or not.

Step 5: Repeat step 3-4, until all-integer valued solutions are recorded.

The above mention method can be represented by an enumeration tree. Each node in the tree represents a sub problem to be evaluated. Each branch of the tree creates a new constraint which is added to the original problem.

3.2 Cutting plane Method

As like as branch and bound algorithm, the cutting plane algorithm also starts at the continuous optimal LP solution. Special constraints (cuts) will be added to the solution space [4]. The iterative procedures of cutting plane method are as follows:

Step 1: Solve the given ILPP ignoring the restriction of integers.

Step 2: If the solution is not integers, find the fractional cut and add new constraint depending on the fractional part of source row and solve using dual simplex method.

Step 3: repeat step 2 until all integer solutions are obtained.

4. PROPOSED TECHNIQUE

An algorithm has been demonstrated for solving ILPPs based on Benders Decomposition by the following steps:

Original P:
$max\ cx + qy, Ax \le b, Tx + Hy \le b, x, y \ge 0$
(variables are x and y, Hy may be a difficult, nonlinear or integer)
Master M(λ^k):
max $cx + z, Ax \le b,\ \lambda^k Tx + z \le \lambda^k b,$
$k = 1,2,,,,,,K, x \ge 0, z\ free$
(variables are x and z (a scalar), λ^k is fixed)
Sub problem S(x^k*):*
min $\lambda(b - T x^k), \lambda H \ge q, \lambda \ge 0$ (variables are λ , x^k is fixed)

Initialize: Set k = 1, pick an x^k (perhaps from maz cx, Ax<=b, x>=0)

Step 1: Solve **S(x^k):** min $\lambda(b - T x^k), \lambda H \ge q, \lambda \ge 0.$ Get $\boldsymbol{\lambda^k}$.
Optimal if $v(\mathrm{S}(x^k)) = v(M(\lambda^k))$.

Step 2: Solve $M(\lambda^k)$: max $cx + z, Ax \le b,\ \lambda^k Tx + z \le \lambda^k b, k = 1,2,,,,,,K,$
$x \ge 0, z\ free$. Get new $\boldsymbol{x^k}$.

Step 3: Let:= k = k+1 and go to Step 1.

5. SOLUTION OF A MODEL PROBLEM USING PROPOSED TECHNIQUE

In this part, one numerical example has been solved manually using proposed technique.

$Minimize \quad 42x_1 + 18x_2 + 33x_3 - 8y_1 - 6y_2 + 2y_3$
$Subject\ to \quad 10x_1 + 8x_2 - 2y_1 - y_2 + y_3 \geq 4$
$5x_1 + 8x_3 - y_1 - y_2 - y_3 \geq 3$
$x_1, x_2, x_3 \in \{0,1\},\ y_1, y_2, y_3 \geq 0$

Solution:

Iteration-1:

Master problem

$Minimize \quad 42x_1 + 18x_2 + 33x_3$
$Subject\ to \quad x_1, x_2, x_3 \in \{0,1\}$

Master problem solution: $x_1 = x_2 = x_3 = 1$
Master value: 93

Primal Sub-problem

$Minimize \quad -8y_1 - 6y_2 + 2y_3$
$Subject\ to \quad -2y_1 - y_2 + y_3 \geq 4 - 10x_1 - 8x_2$
$-y_1 - y_2 - y_3 \geq 3 - 5x_1 - 8x_3$
$y_1, y_2, y_3 \geq 0$

Dual Sub-problem

$Maximize \quad \lambda_1(4 - 10x_1 - 8x_2) + \lambda_2(3 - 5x_1 - 8x_3)$
$= \lambda_1(4 - 10 * 1 - 8 * 1) + \lambda_2(3 - 2 * 1 - 8 * 1)$
$= -14\,\lambda_1 - 10\,\lambda_2$
$Subject\ to \quad -2\lambda_1 - 5\lambda_2 \leq -8$
$-\lambda_1 - \lambda_2 \leq -6$
$\lambda_1 - \lambda_2 \leq 2$
$\lambda_1,\ \lambda_2 \geq 0$

Sub problem solution: $\lambda_1 = 2, \lambda_2 = 4$
Sub problem value: -68

Iteration-2:

Master problem solution: $x_1 = x_2 = x_3 = 0\ and\ z = 20$
Master value: 20
Sub problem solution: $\lambda_1 = 0, \lambda_2 = 8$
Sub problem value: 24

Iteration-3:

Master problem solution:$x_1 = x_2 = 0,\ x_3 = 1\ and\ z = -4$
Master value: 29
Sub problem solution: $\lambda_1 = 4, \lambda_2 = 2$
Sub problem value: 6

Iteration-4:

Master problem solution:$x_1 = 1, x_2 = x_3 = 0\ and\ z = -16$
Master value: 26
Sub problem solution: $\lambda_1 = 0, \lambda_2 = 8$
Sub problem value: -16

Since, At iteration 4 the value of z and sub problem value are same so optimal solution is obtained.

Optimal Solution $x_1 = 1, x_2 = x_3 = 0, y_1 = 2, y_2 = y_3 = 0$ [$y_1, y_2\ and\ y_3\ are\ obtained\ from\ primal\ subproblem$].

6. COMPUTER CODE

One computer code has been developed based on proposed technique to solve ILPPs and solve the problem which is presented in section 5. AMPL [5] has been used to develop the code. Presented code consists of AMPL model file, AMPL data file and AMPL run file. But due to volume only model file and data file have been presented in this paper. If readers are interested then they may contact with the authors.

AMPL model file:

```
# BENDERS DECOMPOSITION #
param k>=1 default 1;                  # iteration


## VARIABLE DECLARETION FOR MASTER ##
param nvm;                             # no. of variables in master
param c {1..nvm};                      # coefficients objective funct
param ncm;                             # no. of constraints in master
param d {1..ncm,1..nvm};               # coefficients in constraints
param b {1..ncm};                      # right hand constants
var xm {1..nvm}>=0 integer;            # variables in master

# # VARIABLE DECLARETION OF SUB-PROGRAM ##
param nvs;                             # no. of variables in subprogram
param a {1..nvs};                      # coeffients of objective function
param nrs;                             # no. of constraints in subprogram
param f {1..nrs,1..nvs};               # coefficients of variables in constr.
param e {1..nrs};                      # right hand constants
var xs {1..nvs}>=0;                    # variables of subprograms

# # VARIABLE DECLAIR PRIMAL SUB-PROGRAM ##
param nvp;                             # no. of variables in primal sub
param g {1..nvp};                      # coeffients of objective function
param ncp;                             # no. of constraints in primal sub
param h {1..ncp,1..nvp};               # coefficients of varia in constra.
param r {1..ncp};                      # right hand constants
var xp {1..nvp}>=0;                    # variables in primal sub problem

##  MASTER ( FOR FIRST ITERATION ) ##
minimize Master_1: sum {j in 1..nvm} c[j]*xm[j];
subject to const_master1 {i in 1..ncm}: sum {j in 1..nvm}
d[i,j]*xm[j] >= b[i];

##  MASTER ( FOR HIGHER ITERATION ) ##
var z;
minimize Master_2: sum {j in 1..nvm-1} c[j]*xm[j]+
                   c[nvm]*z;
subject to const_master2 {i in 1..ncm}: sum {j in 1..nvm-1}
d[i,j]*xm[j]+d[i,nvm]*z >= b[i];

## SUB PROBLEM ##
maximize Sub_v:  sum {j in 1..nvs} a[j]*xs[j];
subject to const_sub {i in 1..nrs}: sum {j in 1..nvs} f[i,j]*xs[j]
                   <= e[i];

## PRIMAL SUB PROBLEM ##
minimize primal_v: sum {j in 1..nvp} g[j]*xp[j];
subject to const_primal {i in 1..ncp}: sum {j in 1..nvp}
h[i,j]*xp[j] >= r[i];
```

AMPL data file:

```
Param  nvm :=  3;
param  c :=
        1     3
        2     18
        3     33;
param ncm :=  1;
param  d:   1  2  3:=
       1    0  0  0 ;

param  b :=
       1    0;
param nvs := 2;
param nrs := 3;
param f:   1   2 :=
       1     -2  -1
param e :=
        1     -8
        2     -6
        3     2;
param nvp := 3;

param g:=
        1     -8
        2     -6
        3     2;
param ncp := 2;

param h:  1   2   3 :=
        1    -2  -1  1
```

```
2    -1  -1                    2    -1  -1 -1;
3     1  -1;
```

AMPL Output System:

Like other software such as FORTRAN, MATHEMATICA, MATHLAB, LINDO etc. AMPL has an intrinsic system to run code. In AMPL model file and data file have to write in different text files. Then one can generate a run file and have to call model and data file in that run file. AMPL has different solvers.

7. SOLUTION USING COMPUTER CODE

The programming output of the problem which is presented in section 5 has been given bellow. The output has developed by using the code which is presented in section 6.

```
iteration = 1
Solve Master:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
1 iterations, objective 93
xm [*] :=
    1   1
    2   1
    3   1;
Solve Sub problem:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
0 iterations, objective -68
xs [*] :=
    1   2
    2   4;
iteration = 2
Solve Master:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
1 iterations, objective 20
xm[1]=0
xm[2]=0
xm[3]=0
z = 20
Solve Sub problem:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
0 iterations, objective 24
xs [*] :=
    1   0
    2   8;
iteration = 3
Solve Master:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
1 iterations, objective 29
xm[1]=0
xm[2]=0
xm[3]=1
z = -4

Solve Sub problem:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
0 iterations, objective 6
xs [*] :=
    1   4
    2   2;
iteration = 4
Solve Master:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
1 iterations, objective 26
xm[1]=1
xm[2]=0
xm[3]=0
z = -16
Solve Sub problem:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
0 iterations, objective -16
xs [*] :=
    1   0
    2   8;
Solve Primal:
MINOS 5.5: ignoring integrality of 1 variables
MINOS 5.5: optimal solution found.
0 iterations, objective -16
xp [*] :=
    1   2
    2   0
    3   0;
Optimal Solution:
os [*] :=
    1   1
    2   0
    3   0
    4   2
    5   0
    6   0;
ov = 26
```

8. RESULT AND DISCUSSION

Table 1. Comparison between the manual output and program output

It. No	Manual Output	Program Output
1	$Master\ solution$: $x_1 = x_2 = x_3 = 1$ $Master\ value$: $M_1(v) = 93$ $Subproblem\ solution$: $\lambda_1 = 2, \lambda_2 = 4$ $Subproblem\ value$: $S_1(v) = -68$	$Master\ solution$: $x_1 = x_2 = x_3 = 1$ $Master\ value$: $M_1(v) = 93$ $Subproblem\ solution$: $\lambda_1 = 2, \lambda_2 = 4$ $Subproblem\ value$: $S_1(v) = -68$
2	$Master\ solution$: $x_1 = x_2 = x_3 = 0$ $Master\ value$: $M_2(v) = 20, z = 20$ $Subproblem\ solution$: $\lambda_1 = 0, \lambda_2 = 8$ $Subproblem\ value$: $S_2(v) = 24$	$Master\ solution$: $x_1 = x_2 = x_3 = 0$ $Master\ value$: $M_2(v) = 20, z = 20$ $Subproblem\ solution$: $\lambda_1 = 0, \lambda_2 = 8$ $Subproblem\ value$: $S_2(v) = 24$
3	$MMaster\ solution$: $x_1 = x_2 = 0, x_3 = 1$ $Master\ value$: $M_3(v) = 29, z = -4$ $Subproblem\ solution$: $\lambda_1 = 4, \lambda_2 = 2$ $Subproblem\ value$: $S_3(v) = 6$	$MMaster\ solution$: $x_1 = x_2 = 0, x_3 = 1$ $Master\ value$: $M_3(v) = 29, z = -4$ $Subproblem\ solution$: $\lambda_1 = 4, \lambda_2 = 2$ $Subproblem\ value$: $S_3(v) = 6$
4	$Master\ solution$:	$Master\ solution$:

	$x_1 = 1, x_2 = x_3 = 0$ *Master value*: $M_4(v) = 26, z = -16$ *Subproblem solution*: $\lambda_1 = 0, \lambda_2 = 8$ *Subproblem value*: $S_4(v) = -16$	$x_1 = 1, x_2 = x_3 = 0$ *Master value*: $M_4(v) = 26, z = -16$ *Subproblem solution*: $\lambda_1 = 0, \lambda_2 = 8$ *Subproblem value*: $S_4(v) = -16$

Form the above table it is shown that the developed computer code gives the same results with manual output.

9. CONVERGENCY TEST OF MASTER VALUE AND SUB PROBLEM VALUES

```
<<Graphics`MultipleListPlot`
 a={{1,0},{2,20},{3,-4},{4,-16}};
s={{1,-68},{2,24},{3,6},{4,-16}};
 MultipleListPlot[a,s,AxesLabel→
{"Iteration No.","Objective
Value"},PlotJoined→ True,PlotLegend→
{value_z,Sub}];
```

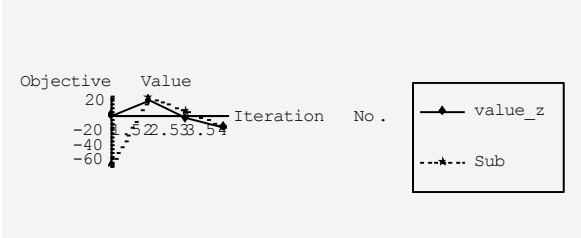

Figure 1. Convergence of the Sub-problem and Master-problem Values [13]

10. CONCLUSION

In this paper a new technique had presented for solving ILPPs. The idea of benders decomposition method had been used for developing this system. By using AMPL one computer code had been developed to solve ILPPs easily. Moreover, the graphical representations had been illustrated to show the Convergence of the master and the sub-problem values using MATHEMATICA. This improved technique will be extended to solve large scale Mixed-Integer Programming. Finally it is noted that the presented decomposition algorithm can be used as an effective tool for solving ILPPs to avoid the laborious calculations using row generation.

11. ACKNOWLEDGEMENT

Our thanks for the experts who have contributed towards the development of this paper.

12. REFERENCES

[1] Hossain, M. I and Hasan, M.B., *An Improved Decomposition Algorithm and Computer Technique For Solving LPs*, IJBAS-IJENS Vol. 11 No: 03.

[2] Das, H. K. and Babul Hasan, M., *An Improved Decomposition Approach and its Computer Technique For Solving Primal Dual LP & LFP Problems*, GANIT, J. Bangladesh mathematical Society Vol. 33, 2013, pp. 65-75.

[3] Sajal chakroborty and M. Babul Hasan, *A Chronic of Analyzing Stochasticity in Multi Period Transportation Problems for Uncertainity*, IJCA Vol. 113 No: 08, January 2016.

[4] Taha. H. A., Operations research: An introduction, 8th Ed. Pearson Preceton hall.

[5] R. Fourer, D. M. Gay and B. W. Kernighan, AMPL, A Modeling Language of Mathematical programming, 2nd Edition.

[6] Winston, W.L. (1994), "*Linear Programming: Applications and Algorithm*", Duxbury press, Belmont, California, U.S.A.

[7] Gupta, P.K., D.S. Hira, 2005. Problems in Operations Research Principles and Solution, S. Chand & Copany LTD., New Delhi-110055, 360-405.

[8] Dantzig, G.B. and P. Wolfe (1961), "*The Decomposition Algorithm for Linear Programming*", Econometrica, **Vol**. 29, No.4.

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

[10] Ravindran, Philips & Solberg (2000), "Operations Research", John Wiley and Sons, Second Edition, New York, U.S.A.

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

[12] Dantzig, G.B. (1963), "*Linear Programming and Extensions*", Princeton University Press, Princeton, U.S.A.

[13] Don, E. (2000), "*Theory and Problems of Mathematica*", Schaum's Outline Series, Mc. GRAW-HILL.