SOLVING THE BINARY INTEGER BI-LEVEL LINEAR PROGRAMMING PROBLEM

Peter M. Hocking

A Thesis Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
Of the Requirements for the Degree of
Master of Arts

Department of Mathematics and Statistics

University of North Carolina at Wilmington

2001

Approved by

Advisory Committee

Chair

Accepted by

Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

This thesis will introduce a historical perspective of the development of work in the field of multi-level linear programming. It will then proceed to extend the theoretical work of the the mixed integer bi-level linear programming problem to encompass the binary integer bi-level linear programming problem. An algorithm will be developed to solve this particular problem using a preference function to determine the choice of branching in a branch and bound tree. Computational results will be compiled and the implications discussed.

ACKNOWLEDGMENTS

The author would like to thank the following people.

Dr. John Karlof for his patience and freely given knowledge throughout the writing of this thesis.

Dr. James Blum for his expert assistance in coding the algorithm in SAS and his never-ending enthusiastic encouragement.

The author would also thank his Graduate School colleagues for their friendly advice at opportune moments.

1 INTRODUCTION

Planning in hierarchical organizations is an interactive process where a central unit (leader) coordinates a lower level unit (follower). This process becomes more complex to implement, coordinate and optimize when the follower is afforded some level of autonomy. In some instances, the objectives of the follower may conflict with those of the leader.

Historically, this interactive process was mirrored in the mathematical programming techniques employed. However, the limitations of the one decision maker criteria [4] used in this type of programming, even when extended to multi-criteria, forced a re-evaluation of the single objective formulation and its attendant techniques. Thus was born multi-level programming and its diverse application areas.

The earlier models developed the "leader/follower" problem. Here the leader controls the decision vector $\mathbf{x} \in X \subseteq R^{n1}$ and the follower controls the decision vector $\mathbf{y} \in Y \subseteq R^{n_2}$. The leader is given first choice and selects an $\mathbf{x} \in \Omega(X) \subseteq X$ to minimize the objective function $\boldsymbol{F}$ where $\boldsymbol{F}$ is also a function of $\mathbf{y}$. The follower then chooses a $\mathbf{y} \in Y \cap \Omega(x)$ to minimize the objective function $\boldsymbol{f}$, where the sets $\Omega(X)$ and $\Omega(x)$ place additional restrictions on the feasible region of the leader and follower, respectively. This leads to the bi-level linear programming problem (BLPP) [1].

This structure is very useful in many commercial and government arenas. A classic, and topical, area is in the way government handles its energy policy. Reducing dependence on imported product would appear desirable, hence government can set import levies, sales taxes, import quotas and, in extreme cases, rationing. Individuals will then decide their consumption according to the resulting pricing and availability. This consumption will then affect the levels of imports, government revenues and price levels. It is this sequential dependence action that makes the use and development of bi-level linear programming obviously applicable in this case and in a wide range of areas.

This idea was further developed in Bard and Falk's 1989 paper [2] where they generated the following mixed integer linear bi-level programming problem.

Let x^1 be an n_{11}-dimensional vector of continuous variables and x^2 be an n_{12}-dimensional vector of discrete variables, where $\mathbf{x} \equiv (x^1, x^2)$ and $n_1 = n_{11} + n_{12}$. Similarly let y^1 be an n_{21}-dimensional vector of continuous variables and y^2 be an n_{22}-dimensional vector of discrete variables, where $\mathbf{y} \equiv (y^1, y^2)$ and $n_2 = n_{21} + n_{22}$.

This leads to

$$\max_{x} \; F(\mathbf{x},\mathbf{y}) = c^{11}x^1 + c^{12}x^2 + d^{11}y^1 + d^{12}y^2 \tag{1a}$$

subject to

$$\mathbf{x} \in X = \{\mathbf{x} : D^1x^1 + D^2x^2 \leq b^1\} \tag{1b}$$

$$\max_{y} \; f(\mathbf{y}) = d^{21}y^1 + d^{22}y^2 \tag{1c}$$

subject to

$$g(\mathbf{x},\mathbf{y}) = A^1x^1 + A^2x^2 + B^1y^1 + B^2y^2 \leq b^2 \tag{1d}$$

$$\mathbf{y} \in Y = \{\mathbf{y} : C^1y^1 + c^2y^2 \leq b^3\} \tag{1e}$$

$$x \geq 0, \quad y \geq 0 \quad x^2, y^2 \text{ integer} \tag{1f}$$

Bard and Falk utilized the following notation and definitions in their work:

BLPP Constraint Region

$\Omega = \{(\mathbf{x},\mathbf{y}) : \mathbf{x} \in X, \; \mathbf{y} \in Y, \; g(\mathbf{x},\mathbf{y}) \leq b^2\}$.

Projection of Ω onto the Leader's Decision Space

$\Omega(X) = \{\mathbf{x} \in X : \exists \; \mathbf{y} \text{ such that } (\mathbf{x},\mathbf{y}) \in \Omega\}$.

Follower's Feasible Region for x ∈ X Fixed

$\Omega(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} \in Y, \; g(\mathbf{x},\mathbf{y}) \leq b^2\}$.

Follower's Rational Reaction Set

$M(\mathbf{x}) = \{\mathbf{y} : arg \max(f(y') : y' \in \Omega(\mathbf{x}))\}$.

Inducible Region

$IR = \{(\mathbf{x},\mathbf{y}) : \mathbf{x} \in \Omega(X),\ \ \mathbf{y} \in M(\mathbf{x})\}.$

In order to make (1) well posed it is assumed that Ω is non-empty and compact, and that for each decision taken by the leader there is some room to move for the follower, or $\Omega(\mathbf{x}) \neq \emptyset$.

Definition 1 *If $\bar{\boldsymbol{y}} \in M(\bar{\boldsymbol{x}})$ then $\bar{\boldsymbol{y}}$ is said to be optimal with respect to $\bar{\boldsymbol{x}}$; such a pair is said to be* bi-level feasible.

Definition 2 *A point (x^*, y^*) is said to be an optimal solution to the BLPP if*
a. (x^*, y^*) is bi-level feasible ; and,
b. for all feasible pairs $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \in IR, \quad F(x^*, y^*) \geq F(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

At this point Bard and Falk postulated several obstacles to the development of algorithms to solve this problem. In their 1990 paper [2] they established three fathoming rules for general mixed integer programming problems. Fathoming in normal linear programming scenarios presents no problems and follows three simple rules.

Rule 1 *The relaxed subproblem has no feasible solution.*

Rule 2 *The solution of the relaxed subproblem is no greater than the value of the incumbent.*

Rule 3 *The solution of the relaxed subproblem is feasible to the original problem.*

In the BLPP, unfortunately, only Rule 1 can be applied with any degree of confidence. Rule 2 needs some strong qualification and Rule 3 must be discarded altogether. Two simple examples will highlight these difficulties.

Example 1

$$\max_{x} \quad F(x, y) \quad = \quad x + 10y$$

$$\begin{array}{rrcl} & \textit{where } y \textit{ solves} & & \\ & \max_y \quad f(x,y) & = & -y \\ \textit{subject to} & -25x + 20y & \leq & 30 \\ & x + 2y & \leq & 10 \\ & 2x - y & \leq & 15 \\ & 2x + 10y & \geq & 15 \\ & x, y & \geq & 0 \\ & x, y \quad \textit{integer} & & \end{array}$$

In this example if the integrality constraints are removed then the relaxed solution is $(x, y) = (8, 1)$ with $F(x, y) = 18$. However the true optimum with the extra requirements of integrality, which should, by fathoming Rule 2 be less than the relaxed solution, is actually $(x^*, y^*) = (2, 2)$ with $F(x^*, y^*) = 22$.
This gives rise to 2 observations.

Observation 1 *The solution of the relaxed BLPP does not provide a valid upper bound on the solution of the mixed integer BLPP.*

Observation 2 *Solutions to the relaxed BLPP that are in the inducible region cannot, in general, be fathomed.*

Example 2

$$\begin{array}{rrcl} & \max_x \quad F(x,y) & = & -x - 2y \\ & \textit{where } y \textit{ solves} & & \\ & \max_y \quad f(x,y) & = & y \\ \textit{subject to} & -x + 2.5y & \leq & 3.75 \\ & x + 2.5y & \geq & 3.75 \end{array}$$

$$\begin{aligned} 2.5x + y &\leq 8.75 \\ x, y &\geq 0 \\ x, y \quad & integer \end{aligned}$$

In this example the BLPP constraint region includes three integer points:
$(2,1)$, $(2,2)$ and $(3,1)$. If the leader chooses $x = 2$, the follower picks $y = 2$ and $F = -6$. If the leader's choice is $x = 3$ then the follower will choose $y = 1$ so $F = -5$. Hence the optimal solution is $(x^*, y^*) = (3,1)$ with $F = -5$.
However in a depth first branch and bound technique Rule 3 would fathom the branch including $(3,1)$ at $(2,1)$ which is not in the inducible region and the optimal solution would never be achieved. This gives rise to the third observation.

Observation 3 *All integer solutions to the relaxed BLPP with some of the follower's variables restricted cannot, in general, be fathomed.*

Development continued and the more work performed, the more the conclusion that the solution to the BLPP was difficult and complex was reinforced. Some, like Bard and Moore [3], worked on algorithms for specific cases, specifically exploiting the follower's Kuhn-Tucker conditions. Hansen et.al [5] focussed on determining necessary optimality conditions expressed in terms of the tightness of the follower's constraints and developing a penalty structure for the branch and bound method.

Vincente et. al. [6] analyzed different discretizations of the the set of variables. Studying the geometry of the feasible set and relating the classes of discrete linear problems to each other, they established equivalences. These equivalences were based on concave penalty functions and this would help to design penalty function methods for the solution of discrete linear programming problems.
They defined three cases to consider:
DCLB: only leader's variables take discrete values

DLB: both leader and follower variables are forced to be integer

CDLB: only follower's variables can take integer values.

In their conclusions they state that DCLB and DLB type problems always have optimal solutions under hypotheses simlilar to the linear case. However, CDLB type problems may, but not necessarily will have an optimal solution.

In their 1990 paper Wen and Yang [7] laid out the general form of their mixed integer two level problem.

$$\max_x \ \{f_1(x^1, x^2) = c^{11}x^1 + c^{12}x^2 : (x^1)\}$$

subject to

$$\max_x \ \{f_2(x^1, x^2) = c^{22}x^2 : (x^2|\hat{x}^1)\}$$

subject to

$$A^1x^1 + A^2x^2 \leq b$$

$$x^1 = \{x^1_1, x^1_2, \dots, x^1_{n1}\}$$

$$x^1_j = \{0, 1\}, \quad j = 1, 2, \dots, n1$$

$$x^2 \geq 0.$$

Using the notations:

$S = \{(x^1, x^2)|A^1x^1 + A^2x^2 \leq b,\ x^1_j \in \{0, 1\},\ j = 1, 2, \dots, n1; x^2 \geq 0\}$

and $W_{f_2}(S) = \{(\hat{x}^1, \hat{x}^2) \in S|f_2(\hat{x}^1, \hat{x}^2)\} = \max\{f_2(x^1, x^2) : (x^2|\hat{x}^1)\}$ This lead to two definitions:

Definition 1 *A point $(\hat{x}^1, \hat{x}^2)$ is said to be feasible to the mixed integer two level linear programming problem if $(\hat{x}^1, \hat{x}^2) \in W_{f_2}(S)$.*

Definition 2 *A point (x^{1*}, x^{2*}) is said to be an optimal solution to the mixed integer two level linear programming problem if both (x^{1*}, x^{2*}) is feasible AND for all feasible points $(\tilde{x}^1, \tilde{x}^2) \in S,\ c^{11}x^{1*} + c^{12}x^{2*} \geq c^{11}\tilde{x}^1 + c^{12}\tilde{x}^2$.*

Utilizing these definitions the authors then constructed a theorem and two lemmas to assist in the development of their algorithm, alleviating some of the concerns previously raised about the viability of the bounding and ability to find an optimal solution.

Their first lemma asserts the following:
Given two linear programming problems of the form

$$\begin{aligned} (P): \quad \max \quad & Z = \sum_{j=1}^{n} c_j x_j \\ st: \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \geq 0, \quad j = 1, 2, \ldots, n \end{aligned}$$

and

$$\begin{aligned} (P^1): \quad \max \quad & Z^1 = \sum_{j=1}^{n} c_j x_j \\ st: \quad & \sum_{j=1}^{n} a_j x_j \leq b + \theta \\ & x_j \geq 0, \quad j = 1, 2, \ldots, n \end{aligned}$$

then $Z^{1*} \leq Z^* + Y^*\theta$. This will be proven in depth later in this thesis.
The second lemma shows that the leader's optimal objective function value for a mixed integer two level linear programming problem, with some of the leader's variables fixed, is less than or equal to the optimal objective function value for the same problem reformatted as a linear programming problem, by removing the follower's objective function. This too will be explored later in the thesis.
Their main theorem is powerful and is stated thus:
If Z_B^* is the optimal objective function value, and Y_B^* is the optimal dual solution of a

problem in this form

$$\begin{aligned}(B):\quad \max\quad & Z_B = \sum_{j=1}^{n2} c_j^{12} x_j^2 \\ st:\quad & \sum_{j=1}^{n2} a_j^2 x_j^2 \leq b \\ & x_j^2 \geq 0, \quad j = 1, 2, \ldots, n2\end{aligned}$$

then

$$Z^U = Z_B^* + \sum_{j \in J_k^+} (c_j^{11} - Y_B^* a_J^1) + \sum_{j \in J_k^0} \max\{(c_j^{11} - Y_B^* a_J^1), 0\}$$

is an upper bound for the leader's objective function.

This thesis will extend the work of Wen and Yang [7] to produce an algorithm to solve the binary bi-level linear programming problem.

In the algorithm developed by Wen and Yang [7], a branch and bound method was employed to establish values for the leader's variables since these were discrete. The algorithm created by Wen and Yang uses the simple expedient of initially proceeding to the bottom of the tree by making all the leader's variables zero. Whilst this does ensure a feasible solution, if the original relaxed problem is feasible, it does not seem efficient in its approach to finding the optimal feasible solution. The proposed algorithm will impose a preferential choice, in the branch and bound tree, based on calculated upper bounds for the leader's variables $\{x_j^1\}$.

2 BINARY GENERAL FORM AND NOTATION

In their 1990 paper Wen and Yang laid out the general form of their mixed integer two level problem.[7]

The binary two-level programming problem adaptation of their mixed integer two level programming problem is presented in its general form as follows:

$$\max_{x} \; \{f_1(x^1, x^2) = c^{11}x^1 + c^{12}x^2 : (x^1)\} \tag{2a}$$

subject to

$$\max_{x} \; \{f_2(x^1, x^2) = c^{22}x^2 : (x^2|\hat{x}^1)\} \tag{2b}$$

subject to

$$A^1x^1 + A^2x^2 \leq b^2 \tag{2c}$$

$$x^1 = \{x^1_1, x^1_2, \ldots, x^1_{n1}\} \tag{2d}$$

$$x^2 = \{x^2_1, x^2_2, \ldots, x^2_{n2}\} \tag{2e}$$

$$x^1_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n1 \tag{2f}$$

$$x^2_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n2 \tag{2g}$$

where

$\max_x \; \{f_1(x^1, x^2) = c^{11}x^1 + c^{12}x^2 : (x^1)\}$ denotes the maximum of f_1 over x^1, x^2 but only x^1 can be set at the leader's problem level.

$\max_x \; \{f_2(x^1, x^2) = c^{22}x^2 : (x^2|\hat{x^1})\}$ denotes the maximum of f_2 over x^2 for a fixed value of $\hat{x}^1$

and

x^1 is a $n1 \times 1$ vector of decision variables controlled by the leader;

x^2 is a $n2 \times 1$ vector of decision variables controlled by the follower;

A^1 is a $m \times n1$ matrix of coefficients for the leader's variables;

A^2 is a $m \times n2$ matrix of coefficients for the follower's variables;

c^{11} is a $1 \times n1$ vector of cost/profit coefficients of leader's variables in the leader's objective function f_1;

c^{12} is a $1 \times n2$ vector of cost/profit coefficients of follower's variables in the leader's objective function f_1;

c^{22} is a $1 \times n2$ vector of cost/profit coefficients of follower's variables in the follower's objective function f_2;

b is a $m \times 1$ vector of resource capacity of the system.

This general formulation will be manipulated in accordance with the modified Lemmas and Theorem to allow the development of the algorithm to solve binary bi-level linear programming problems. At various points in the reformulation of the problem the objective functions for the leader, (2a), and the follower, (2b), will be removed as a relaxation of the general problem and the theorem applied. This removal of one of the objective functions, along with the specification of some of the values of the leader's variables will reduce the problem to a binary linear programming problem.

3 THE BINARY PROBLEM FORMULATION

In Wen and Yang's [7] paper they utilized a notation for the value of the leader's variables at any particular level in the branch and bound tree. This useful notation will be kept throughout the course of this thesis. It allows the flexibility to divide the values of the leader's variables into three distinct sets. These separate sets will allow control over the process of determining the leader's variables' values. The sets will also contribute to the calculation of the preference indicator for deciding the initial choice of branching in the branch and bound tree. This indicator will also be used in the fathoming of the branches later in the algorithm.

This notation is defined as follows:

k: the order number of generated node in a branch-and-bound tree:

$J_k^0 = \{j | x_j^1$ is a free binary variable, $j = 1, 2, \ldots, n1\}$;

$J_k^+ = \{j | x_j^1$ is a fixed at 1, $j = 1, 2, \ldots, n1\}$;

$J_k^- = \{j | x_j^1$ is a fixed at 0, $j = 1, 2, \ldots, n1\}$;

This allows the formulation of the binary problem (TP_f) in terms of the fixing of some of the variable values as follows:

$$(TP_f):\quad \max_x\ f_1 = \sum_{j\in J_k^0} c_j^{11}x_j^1 + \sum_{j\in J_k^+} c_j^{11} + \sum_{i=1}^{n2} c_i^{12}x_i^2 \tag{3a}$$

subject to

$$\max_x\ f_2 = \sum_{i=1}^{n2} c^{22}x^2 \tag{3b}$$

subject to

$$\sum_{j\in J_k^0} a_j^1 x_j^1 + \sum_{i=1}^{n2} a^2x^2 \leq b - \sum_{j\in J_k^+} a_j^1 \tag{3c}$$

$$x_j^1 \in \{0,1\},\quad j\in J_k^0 \tag{3d}$$

$$x_i^2 \in \{0,1\},\quad i = 1,2,\ldots,n2 \tag{3e}$$

The effect of fixing some of the values of the leader's variables can clearly be seen in equation (3a). Here, the three distinct sets established earlier are utilized. The x_j^1 where $j \in J_k^-$, i.e. where $x_j^1 = 0$ have been removed from the leader's objective function. The splitting of the leader's variables is more significant in equation (3c) where it is seen that the a_j^1 associated with the $x_j^1 \in J_k^+$ move to the resources side of the constraint equation. The importance of this will become apparent in later sections.

Relaxing the TP_f by removing the follower's objective function creates a problem denoted as P_f. It appears, after minor rearrangement, as follows:

$$(P_f):\quad \max_x\ g = \sum_{j\in J_k^0} c_j^{11}x_j^1 + \sum_{j\in J_k^+} c_j^{11} + \sum_{i=1}^{n2} c_i^{12}x_i^2 \tag{4a}$$

subject to

$$\sum_{i=1}^{n2} a^2x^2 \leq b - \sum_{j\in J_k^+} a_j^1 - \sum_{j\in J_k^0} a_j^1x_j^1 \tag{4b}$$

$$x_j^1 \in \{0,1\},\quad j\in J_k^0 \tag{4c}$$

$$x_i^2 \in \{0,1\}, \quad i = 1,2,\ldots,n2 \tag{4d}$$

4 BOUNDING THEOREM AND LEMMAS

In their paper Wen and Yang proved the following:

Lemma 1 *[7]*

Given two linear programming problems:

$$\begin{aligned} (P):\quad max \quad & Z = \sum_{j=1}^{n} c_j x_j \\ st: \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \geq 0, \quad j = 1,2,\ldots,n \end{aligned}$$

and

$$\begin{aligned} (P^1):\quad max \quad & Z^1 = \sum_{j=1}^{n} c_j x_j \\ st: \quad & \sum_{j=1}^{n} a_j x_j \leq b + \theta \\ & x_j \geq 0, \quad j = 1,2,\ldots,n \end{aligned}$$

where

a_j is the jth. column vector of the $m \times n$ matrix, A;

θ is a $m \times 1$ parameter vector.

Then, if

Z^* is the optimal objective value of P;

Y^* is a $1 \times m$ vector, denoting the dual optimal solution of P;

Z^{1*} is the optimal objective value of P^1; and

Y^{1*} is a $1 \times m$ vector, denoting the dual optimal solution of P^1,

then

$Z^{1*} \leq Z^* + Y^*\theta$.

Proof: Let Y^* be the optimal dual solution of P. Thus Y^* is also a feasible dual solution of P^1. This gives rise to:

$$\begin{aligned} (Dual\ \ of\ \ P):\ \ & min\ \ Yb \\ where\ \ & YA \geq c \\ & Y \geq 0 \end{aligned}$$

and

$$\begin{aligned} (Dual\ \ of\ \ P^1):\ \ & min\ \ Y(b+\theta) \\ where\ \ & YA \geq c \\ & Y \geq 0. \end{aligned}$$

Returning to the primal:

$$\begin{aligned} Z^{1*} &= Y^{1*}(b+\theta) && \leq Y^*(b+\theta) \\ & && = Y^*b + Y^*\theta \\ & && = Z^* + Y^*\theta. \quad \blacksquare \end{aligned}$$

This Lemma will play a central role in the development of Theorem 1, which follows.

Lemma 2 *The optimal value of the leader's objective function, f_1^*, in the TP_f is less than or equal to the optimal objective function value, g^*, in the P_f problem.*

Proof: This proof is obvious, the solution space of the TP_f is contained in the solution space of the P_f. This evolves from the fact that the P_f is the TP_f without the follower's objective function and is thus less constrained. Hence, given this relationship it is clear that $f_1^* \leq g^*$. $\blacksquare$

Theorem 1 *Consider the following problem denoted problem B:*

$$
\begin{aligned}
(B): \quad max \quad & Z_B = \sum_{i \in I_k^0} c_i^{12} x_i^2 \\
st: \quad & \sum_{i=1}^{n2} a_i^2 x_i^2 \leq b \\
& x_i^2 \geq 0, \quad x_i^2 \leq 1 \quad i = 1, 2, \ldots, n2
\end{aligned}
$$

Let Z_B^* be the optimal objective function value for problem B above. Also let Y^* be the optimal dual solution of problem B. Then an upper bound, Z^U, is established for the leader's objective function value in problem TP_f where:

$$Z^U = Z_B^* + \sum_{j \in J_k^+} (c_j^{11} - Y_B^* a_J^1) + \sum_{j \in J_k^0} max\{(c_j^{11} - Y_B^* a_J^1), 0\} \tag{5}$$

That is $Z^U \leq f_1^*$.

Proof: Recall that P_f is:

$$
\begin{aligned}
(P_f): max \quad g = & \sum_{j \in J_k^0} c_j^{11} x_j^1 + \sum_{j \in J_k^+} c_j^{11} + \sum_{i=1}^{n2} c_i^{12} x_i^2 \\
st: \quad & \sum_{i=1}^{n2} a^2 x^2 \leq b - \sum_{j \in J_k^+} a_j^1 - \sum_{j \in J_k^0} a_j^1 x_j^1 \\
& x_j^1 \in \{0, 1\}, \quad j \in J_k^0 \\
& x_i^2 \in \{0, 1\}, \quad i = 1, 2, \ldots, n2
\end{aligned}
$$

P_f is relaxed by replacing the conditional constraint $\{x_i^2 \in \{0,1\},\ i = 1, 2, \ldots, n2\}$ by $\{x_i^2 \leq 1,\ x_i^2 \geq 0,\ i = 1, 2, \ldots, n2\}$.

This relaxation produces a problem denoted as LP_f.

$$(LP_f): max \quad g = \sum_{i=1}^{n2} c_i^{12} x_i^2 + \overbrace{\sum_{j \in J_k^0} c_j^{11} x_j^1 + \sum_{j \in J_k^+} c_j^{11}}^{K}$$

$$st: \quad \sum_{i=1}^{n2} a^2 x^2 \leq b - \overbrace{\sum_{j\in J_k^+} a_j^1 - \sum_{j\in J_k^0} a_j^1 x_j^1}^{\theta}$$
$$x_j^1 \in \{0,1\}, \quad j \in J_k^0$$
$$x_i^2 \leq 1, \quad x_i^2 \geq 0 \quad i = 1, 2, \ldots, n2$$

Let g^* to be the optimal objective function of the above LP_f with optimal values $\{x_j^{1*}\}$ of the variables $\{x_j^1\}$. Then we obtain the following linear programming problem.

$$(LP_f^{'}) : max \quad g = \sum_{i=1}^{n2} c_i^{12} x_i^2 + K^{'}$$
$$st: \quad \sum_{i=1}^{n2} a^2 x^2 \leq b - \theta^{'}$$
$$x_i^2 \leq 1, \quad x_i^2 \geq 0 \quad i = 1, 2, \ldots, n2$$

where $K^{'}$ is a constant determined by evaluating $\sum_{j\in J_k^0} c_j^{11} x_j^1 + \sum_{j\in J_k^+} c_j^{11}$ using the values of $\{x_j^{1*}\}$ and $\theta^{'}$ is similarly a constant calculated from $\sum_{j\in J_k^+} a_j^1 + \sum_{j\in J_k^0} a_j^1 x_j^1$ once again using $\{x_j^{1*}\}$.

Then by applying *Lemma* 1 to $LP_f{'}$ and B:

$$g^* - K^{'} \leq Z_B^* + Y_B^* \theta^{'}$$
$$\begin{aligned} g^* &\leq Z_B * + \sum_{j\in J_k^0} c_j^{11} x_j^{1*} + \sum_{j\in J_k^+} c_j^{11} - Y_B^* (\sum_{j\in J_k^+} a_j^1 + \sum_{j\in J_k^0} a_j^1 x_j^{1*}) \\ &= Z_B^* + \sum_{j\in J_k^+} (c_j^{11} - Y_B^* a_j^1) + \sum_{j\in J_k^0} (c_j^{11} - Y_B^* a_j^1) x_j^{1*} \\ &\leq Z_B^* + \sum_{j\in J_k^+} (c_j^{11} - Y_B^* a_j^1) + \sum_{j\in J_k^0} \max\{c_j^{11} - Y_B^* a_j^1, 0\} \end{aligned}$$

Hence $g^* \leq Z^U$.

Now applying *Lemma* 2, clearly $f_1^* \leq g^*$ and so also $f_1^* \leq Z^U$. ■

Re-examining the proof from the dual of the binary problem:

Let π^* be the optimal solution of the Dual of $LP_f{'}$ then,

$$g^* - K^{'} = \pi^* \begin{pmatrix} b - \theta' \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

$$g^* - K^{'} \leq Y_B^* \begin{pmatrix} b - \theta' \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

since Y_B^* is feasible to the dual of $LP_f{}'$

$$g^* - K^{'} \leq Z_B^* + Y_B^* \begin{pmatrix} -\theta' \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

So,

$$g^* \leq Z_B^* + \sum_{j \in J_k^0} c_j^{11} x_j^{1*} + \sum_{j \in J_k^+} c_j^{11} - Y_B^* \begin{pmatrix} \sum_{j \in J_k^+} a_j^1 + \sum_{j \in J_k^0} a_j^1 x_j^{1*} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\leq Z_B^* + \sum_{j \in J_k^+} (c_j^{11} - Y_B^* \begin{pmatrix} a_j^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}) + \sum_{j \in J_k^0} (c_j^{11} - Y_B^* \begin{pmatrix} a_j^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}) x_j^{1*}$$

$$\leq Z_B^* + \sum_{j \in J_k^+} (c_j^{11} - Y_B^* \begin{pmatrix} a_j^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}) + \sum_{j \in J_k^0} \max\{c_j^{11} - Y_B^* \begin{pmatrix} a_j^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, 0\}$$

Again applying *Lemma* 2 it is once again shown that $TP_f^* \leq g^* \leq Z^U$. ■

At this point the tools have been established to create an algorithm to solve the binary bi-level linear programming problem.

5 ALGORITHM

The algorithm depends heavily on the preceding lemmas and theorem. Especially the relaxation of the problem from a two-level problem to a simple binary LP, which is easily and quickly solved compared to solving a complex bi-level linear programming problem. Establishing the relaxed problems is the first priority of the algorithm and is completed in steps 1 and 2. This establishes lower bounds for the solution with all leader's variables set to zero, in both the leader's objective function and the follower's objective function.

At this point it is of interest, perhaps, to note that the follower's objective function does not contain any leader's variables. It has not in any of the preceding formulations. Since the leader's variables will be selected before the follower performs his/her optimization, any leader's variable and attendant coefficient in the follower's objective function will reduce to a constant in the objective function at the time of optimization and hence will not affect that optimization. This allows the ignoring of the leader's variables in the formulation of the follower's objective function.

Step 1 *Initialization*

$$\begin{aligned}
&N = 0, k = 0\\
&\ \textit{N is a place-keeper of the current level in the tree}\\
&\ \textit{k is the counter for evaluated nodes}\\
&J_k^0 = \{1, 2, \ldots, n1\}, J_k^+ = J_k^- = \emptyset\\
&T_j = 0 \ \ j = 1, 2, \ldots, n1
\end{aligned}$$

This indicates that all the leader's variables are free.

Solve problem F:

$$\begin{aligned}
(F:)\ \ \max\ &\sum_{j=1}^{n2} c_j^{22} x_j^2\\
st:\ &\sum_{j=1}^{n2} a_j^2 x_j^2 \leq b\\
&x_j^2 \in \{0,1\}\ \ j = 1, 2, \ldots, n2
\end{aligned}$$

Let the optimal solution $\rightarrow Z^*,\ x^{2*},\ x^{1*} = (\overbrace{0,0,0,\ldots,0}^{n1})$ *where* $Z*$ *is the value of the leader's objective function* f^**, evaluated using the values of* x^{1*} *and* x^{2*}*.*

Step 2 *Relaxed Solution* $x_j^2 \geq 0, \quad x_j^2 \leq 1 \rightarrow Z_B^*,\ Y_B^*$

Solve problem B:

$$\begin{aligned}
(B):\ \ max\ \ & Z_B = \sum_{i \in J_k^0} c_j^{12} x_j^2\\
st:\ \ &\sum_{j=1}^{n2} a_j^2 x_j^2 \leq b\\
&x_j^2 \geq 0, \quad x_j^2 \leq 1 \quad j = 1, 2, \ldots, n2
\end{aligned}$$

This problem results in Z_B^**, the optimal objective function value and* Y_B^**, the optimal dual solution.*

Calculate $H(j) = c_j^{11} - Y_B^* a_j^1,\ j = 1, 2, \ldots, n1.$

In an attempt to maximize the efficiency of finding the optimal solution, some type of penalty, or preference, on the path to follow to the initial leaf at the bottom of the tree needed to be established. This is the function of step 3, the branching step. The value Z_N^U is determined by finding the largest of Z_N^{U+} and Z_N^{U-}. The bound Z_N^{U+} is denoted as the calculation of Z_N^U with $x_N^1 = 1$ and, similarly Z_N^{U-} with $x_N^1 = 0$.

So, Z_N^U is dependent on which of these upper bound quantities is largest, or $Z_N^U = \max\{Z_N^{U+}, Z_N^{U-}\}$. This should help to determine the optimal solution more rapidly, although it must be said that this is not always the case. The validity of this decision will be noted and discussed in the section on computational results.

Step 3 *Branching*

From Theorem 1, $Z^U = Z_B^* + \sum_{j \in J_k^+} H(j) + \sum_{j \in J_k^0} \max\{H(j), 0\}.$

Let $S = \sum_{j \in J_k^+} H(j)$ *and let* $W = \sum_{j \in J_k^0} \max\{H(j), 0\}$

and $N = N + 1 \quad k = k + 1$.

Calculate the upper bound of the leader's objective function Z_N^U *for the branch down-tree from the previous,* $(k-1)$ *for* $x_N^1 = 0$. *This will be denoted as* Z_N^{U-} .

```
Loop 1: Set $S = W = 0$
        For $i = 1$ to $n1$
          If $x_i^1 = 1$
            $S = S + H(i)$
          else
          if $x_i^1 = \emptyset$
            $W = W + max\{H(i), 0\}$
          else
        end for loop
        end loop
```

$Z_N^{U-} = Z_B^* + S + W$

Calculate Z_N^{U+}, the branch where $x_N^1 = 1$.

This is achieved in a very similar manner to the calculation of Z_N^{U-}. Set $x_N^1 = 1$ then execute Loop 1 and finally $Z_N^{U+} = Z_B^* + S + W$.

The decision resulting from the calculation of both Z_N^U's is:

If $Z_N^{U+} \geq Z_N^{U-}$ then the upper bound $Z^U = Z_N^{U+}$, $x_N^1 = 1$ and $T_N = T_N + 1$, otherwise $Z^U = Z_N^{U-}$, $x_N^1 = 0$ and again $T_N = T_N + 1$ (T_N is a counter that registers the number of branches evaluated down-tree from any node in a given iteration of the looping of the algorithm.

As part of the iterative process the upper bounds need to be checked against the current upper bound on the objective function, Z^*. If the upper bound on that particular branch is not greater than the current best solution then that branch may be fathomed.

Step 4 *Optimality Check*

If $Z^U \leq Z^*$ then set $T_N = 2$, go to Step 6.

else go to Step 5.

The next step requires that if the algorithm has arrived at a node at the bottom of the tree then problem L, that is the follower's problem with the values of x_j^1 drawn from the tree at that node, needs to be solved and it's feasibility checked and the solution, Z^L compared to the current best solution.

Step 5 *Calculate Feasible Solutions*

```
        If $N \neq n1$ then go to Step 3
            else output $x^{1L} \rightarrow$ Solve problem L
        where
        (L:) max $\sum_{j=1}^{n2} c_j^{22} x_j^2$
            st: $\sum_{j=1}^{n2} a_j^2 x_j^2 \leq b - \sum_{j \in J_k^+} a_j^2$
            $x_j^2 \in \{0,1\}\ j = 1, 2, \ldots, n2$
```

Solving problem L above gives rise to x^{2*}

Let Z^L *be the leader's objective function value evaluated using* x^{2L} *and* x^{1L}.

If $Z^L > Z^*$ *AND problem L is feasible, then update* Z^*, x^{2*} *and* x^{1*} *from* Z^L, x^{2L} *and* x^{1L} *respectively , then go to Step 6.*

Else go to Step 6.

The algorithm can now proceed back up the tree, examining branches and their upper bounds along the way. Each upper bound compared to the current best solution to determine whether the branch can be fathomed or must be considered further, to extract another leaf of the tree. This is performed in the next step.

Step 6 *Backtracking*

```
If $T_N = 2$ then set
    $T_N = 0,\ x_N^1 = \emptyset, N = N - 1$
    If $N = 0$ go to Step 7.
    else go to Step 6.
else $T_N = T_N + 1$
If $x_N^1 = 0$ then $Z^U = Z^{U+},\ x_N^1 = 1,$ go to Step 4.
else $Z^U = Z_N^{U-},\ x_N^1 = 0$ go to Step 4.
```

All that remains is to terminate the process at the point where all viable branches and leaves have been utilized.

Step 7 *Terminate*

Stop execution of algorithm and output the solution.

To illustrate this algorithm a simple numerical example is formulated.

Example 3

$$
\begin{aligned}
&\max\ 15x_1^1 + 2x_2^1 + 20x_3^1 + 10x_4^1 + 10x_1^2 + 15x_2^2 + 20x_3^2 + 5x_4^2 + 12x_5^2 \\
&\text{(st:)} \\
&\quad \max\ 5x_1^2 + 3x_2^2 + 8x_3^2 + 4x_4^2 + x_5^2 \\
&\text{(st:)} \\
&\\
&\qquad 6x_1^1 + 5x_2^1 + 10x_3^1 + 12x_4^1 + 6x_1^2 + 3x_2^2 + 9x_3^2 + 2x_4^2 + 2x_5^2 \leq 12 \\
&\qquad 2x_1^1 + 4x_2^1 + 13x_3^1 + 7x_4^1 + 5x_1^2 + x_2^2 + 3x_3^2 + 3x_4^2 + x_5^2 \leq 19 \\
&\qquad 3x_1^1 + 8x_2^1 + 9x_3^1 + 9x_4^1 + 10x_1^2 + 5x_2^2 + 6x_3^2 + 4x_4^2 + 6x_5^2 \leq 15 \\
&\qquad 4x_1^1 + 3x_2^1 + 12x_3^1 + 14x_4^1 + 4x_1^2 + 3x_2^2 + 5x_3^2 + x_4^2 + 6x_5^2 \leq 30 \\
&\qquad\quad x_j^1 \in \{0,1\}\ \ j = 1,2,3,4 \\
&\qquad\quad x_i^2 \in \{0,1\}\ \ i = 1,2,3,4,5
\end{aligned}
$$

In the initialization phase both problems F and B are solved to obtain the initial optimal solution Z^* and its attendant followers values x^{2*}, along with the optimal dual and primal solutions to B, Y_B^* and Z_B^* respectively. The execution of the algorithm, for the numerical example above, is as follows:

Step 1: Solve F:

$$\text{F: max } 5x_1^2 + 3x_2^2 + 8x_3^2 + 4x_4^2 + x_5^2$$

(st:)

$$6x_1^2 + 3x_2^2 + 9x_3^2 + 2x_4^2 + 2x_5^2 \leq 12$$
$$5x_1^2 + x_2^2 + 3x_3^2 + 3x_4^2 + x_5^2 \leq 19$$
$$10x_1^2 + 5x_2^2 + 6x_3^2 + 4x_4^2 + 6x_5^2 \leq 15$$
$$4x_1^2 + 3x_2^2 + 5x_3^2 + x_4^2 + 6x_5^2 \leq 30$$
$$x_i^2 \in \{0,1\} \;\; i = 1,2,3,4,5$$

This gives $x^{2*} = (0,0,1,1,0)$ as the optimal solution of problem F. Applying these values in the leader's objective function, with all x^1's zero yields $Z^* = 25$.

Step 2: Solve B and calculate

$$\text{B: max } 10x_1^2 + 15x_2^2 + 20x_3^2 + 5x_4^2 + 12x_5^2$$

(st:)

$$6x_1^2 + 3x_2^2 + 9x_3^2 + 2x_4^2 + 2x_5^2 \leq 12$$
$$5x_1^2 + x_2^2 + 3x_3^2 + 3x_4^2 + x_5^2 \leq 19$$
$$10x_1^2 + 5x_2^2 + 6x_3^2 + 4x_4^2 + 6x_5^2 \leq 15$$
$$4x_1^2 + 3x_2^2 + 5x_3^2 + x_4^2 + 6x_5^2 \leq 30$$
$$x_i^2 \geq 0, \;\; x_i^2 \leq 1, \;\; i = 1,2,3,4,5$$

The solution to problem B is:

$$Z_B^* = 41.476196 \quad Y_B^* = \begin{pmatrix} 1.142857 \\ 0 \\ 1.619048 \\ 0 \\ 0 \\ 3.476191 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Recall the construction of $H(j)$., the values for the above problem are:

$$\begin{aligned} H(1) &= c_1^{11} - Y_B^* a_1^1 \\ &= 3.285714. \end{aligned}$$

Similarly, $H(2) = -16.7 \quad H(3) = -6 \quad H(4) = -18.3$

Step 3: Branching

The first choice facing the algorithm is down which branch will it proceed, $x_1^1 = 0$ or $x_1^1 = 1$. The choice is made dependent on the relative values of the upper bounds for each branch. In this particular case under examination these values are $Z_1^{U-} = 41.476196$ and $Z_1^{U+} = 44.761910$. At this point it would be a useful exercise to show the development of these numbers.

Now consider that if $x_1^1 = 0$ then $J_1^0 = \{2, 3, 4\}$ and $J_1^+ = \emptyset$. Since all $H(j)$'s are negative excepting $H(1)$ and $Z^U = Z_B^* + \sum_{j \in J_k^+} H(j) + \sum_{j \in J_k^0} \max\{H(j), 0\}$ it is clear that Z_1^{U-} will simply equal $Z_B^* = 41.476196$.

Similarly, Z_1^{U+} will be $Z_B^* + H(1)$ which result in a value of 44.761910.

The first nodes will now appear as in Fig.1.

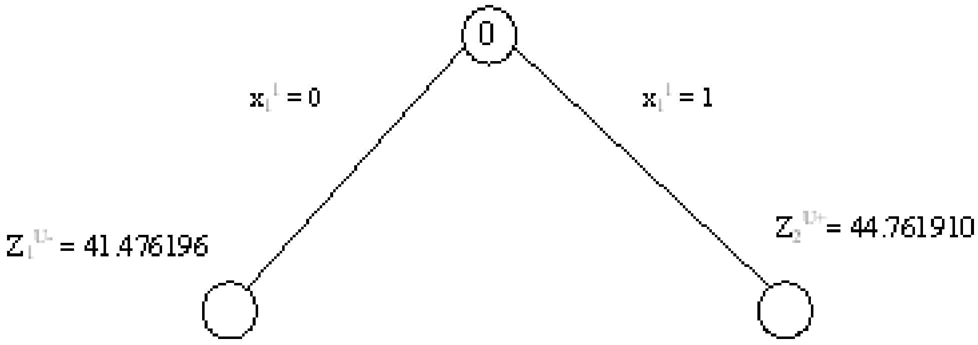

Figure 1: The first decision based on Z^U's.

Continuing, the algorithm will select all x_j^1's according to the highest upper bound on the leader's objective function until the bottom of the branch and bound tree is reached, at which point the tree appears as in Fig.2.

Step 5:

Now that all x_j^1's have been assigned a value of either 0 or 1, i.e. the algorithm has settled on a leaf of the tree, a solution to the follower's problem can be achieved using binary integer linear programming methodology. Using $x^{1L} = (1, 0, 0, 0)$ the integer linear programming problem L, in the example, now becomes:

$$\max \; 5x_1^2 + 3x_2^2 + 8x_3^2 + 4x_4^2 + x_5^2$$

$$\begin{aligned} &\text{(st:)} \\ &6x_1^2 + 3x_2^2 + 9x_3^2 + 2x_4^2 + 2x_5^2 \leq 6 \\ &5x_1^2 + x_2^2 + 3x_3^2 + 3x_4^2 + x_5^2 \leq 17 \\ &10x_1^2 + 5x_2^2 + 6x_3^2 + 4x_4^2 + 6x_5^2 \leq 12 \\ &4x_1^2 + 3x_2^2 + 5x_3^2 + x_4^2 + 6x_5^2 \leq 26 \\ &x_i^2 \in \{0, 1\} \;\; i = 1, 2, 3, 4, 5 \end{aligned}$$

Solving this binary linear programming problem yields:

$Z^L = 35$ & $x^{2L} = (0, 1, 0, 1, 0)$ where once again Z^L is the calculation resulting from the

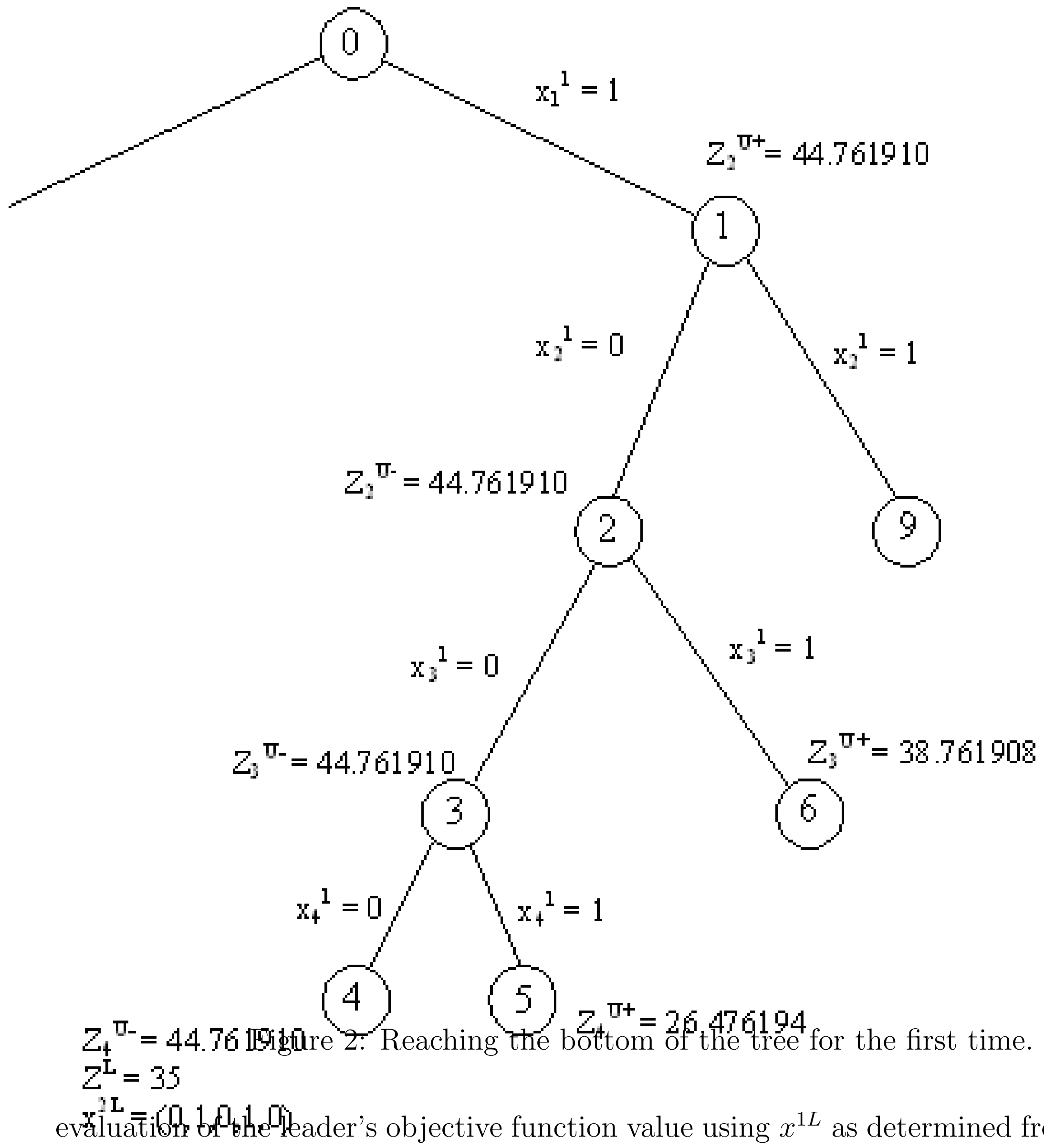

Figure 2: Reaching the bottom of the tree for the first time.

evaluation of the leader's objective function value using x^{1L} as determined from the branch and bound tree and x^{2L}, obtained from the solution of the above problem L.

Since this value of the leader's objective function is larger than the current Z^* (25) an update is performed and $Z^* = 35$, $x^{1*} = (1, 0, 0, 0)$ and $x^{2*} = (0, 1, 0, 1, 0)$. At this point in the algorithm the control variables are $N = 4$, $k = 4$ and $T_3 = 1$.

Step 6:

The backtracking can now take place. It will examine the node associated with $x_4^1 = 1$ and conclude that since $Z_4^{U+} = 26.476194$, which is less than the current Z^* of 35, that this node, although it is a leaf in the tree, does not require formulating into a linear programming problem and subsequent solving. This will eliminate many time-consuming solutions of LP's at the bottom of the tree. This is a great saving, but not the most significant saving. The constant comparison to the best current solution at all nodal decision points will allow the fathoming of many branches higher up the tree, eliminating the need to even proceed to the leaves at the bottom of those branches. An example of this can be seen in Fig.3. At node 9 the value of Z_2^{U+} is only 28.09524. Clearly, there is no need to proceed any further down that branch since it is less than the current best solution and the upper bound would represent the best solution possible down that branch. The full tree as seen in Fig.3 indicates that only 18 of the 30 nodes were considered, and only 4 of the possible 16 leaves were formulated into LP's. This measure will be further discussed in the computational results section.

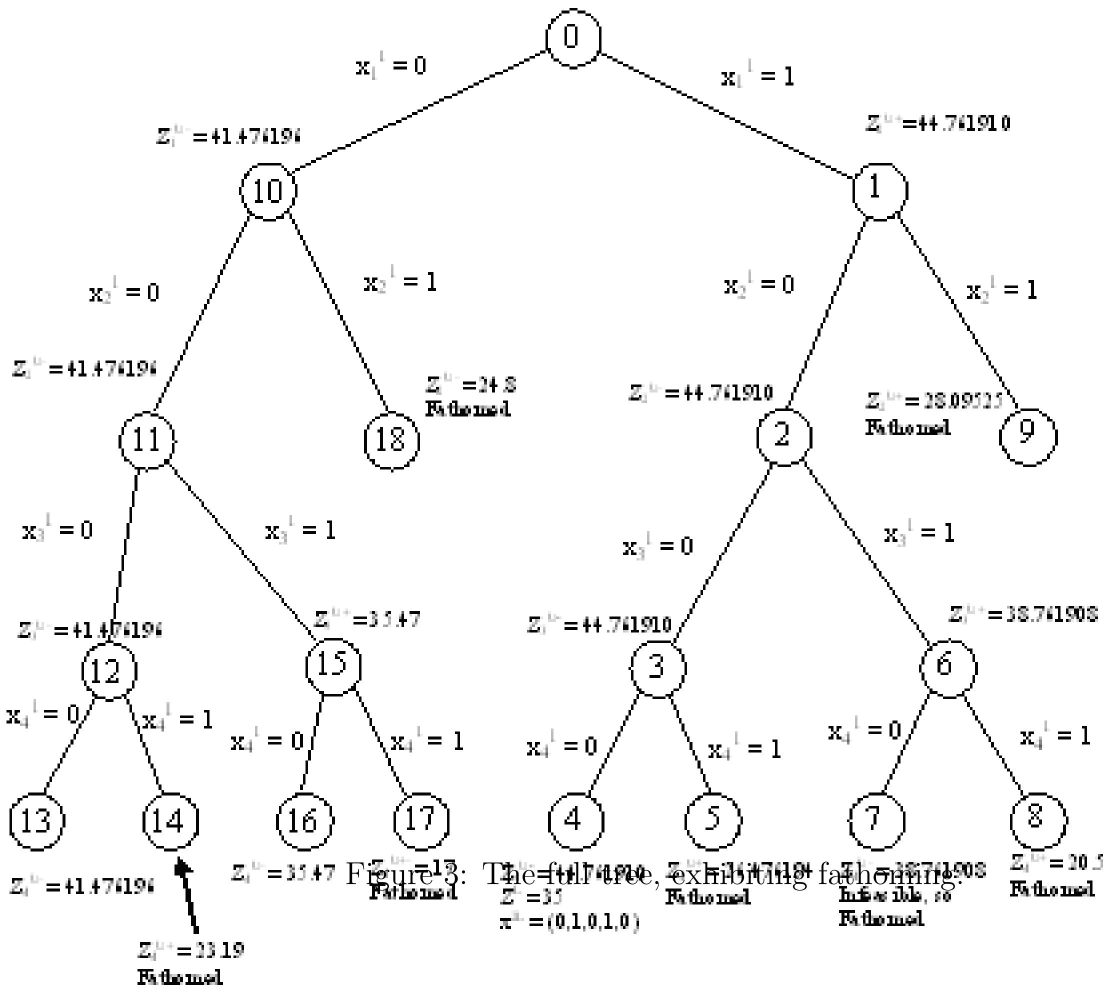

Figure 3: The full tree, exhibiting fathoming.

6 COMPUTATIONAL RESULTS

To evaluate the results of the algorithm it was coded into a SAS program, see Appendix A. Bi-level problems were constructed randomly using the following guidelines.

The leader's objective function variable coefficients were established randomly between limits of -30 to +30. The follower's objective function variable coefficients were placed between -12 and +12. The constraint matrix coefficients were all between -18 and +18 and the b_j, or resource values were restricted to be within the range 0.5 to 0.75 of the sum of the a_j for the j^{th} constraint.

In both table 1 and table 2 the column headers represent

evaluated nodes	=	number of nodes where an upper bound was established as a percentage of total nodes in the tree.
lpcalls	=	number of L problems solved as a percentage of leaves in the tree.
kstar	=	the node number where the optimal solution was obtained as a percentage of nodes in the tree.

In Table 1, 10 randomly constructed problems were solved for each problem type, a combination of $n1 = 5, 8, 10$ and $n2 = 5, 8, 10$, i.e. 5×5, 5×8, 5×10, 8×5, 8×8, 8×10, 10×5 and 10×10. The reader will note that some possible combinations were not included in the list or the table. It was deemed these combinations would not serve to illuminate the results further and thus were omitted.

The results in Table 2 are from randomly constructing 100 problems for each problem type. This set of computations was mainly performed to check the statistical validity of the results in Table 1. A larger sample size would be deemed statistically more significant. The results clearly show the validity of the 10 sample problems, confirming them by their non-significant variation in percentages.

n1	n2	evaluated nodes	total nodes	lpcalls	leaves	kstar
5	5	55%	62	39%	32	27%
5	8	72%	62	62%	32	36%
5	10	75%	62	73%	32	34%
8	5	37%	510	23%	256	13%
8	8	43%	510	31%	256	25%
8	10	64%	510	58%	256	30%
10	5	15%	2046	7%	1024	4%
10	10	51%	2046	41%	1024	11%

Table 1: Results of 10 samples for each $n1 \times n2$ problems

n1	n2	evaluated nodes	total nodes	lpcalls	leaves	kstar
5	5	51%	62	36%	32	30%
5	8	69%	62	57%	32	35%
5	10	71%	62	64%	32	33%
8	5	23%	510	14%	256	12%
8	8	49%	510	37%	256	17%
8	10	57%	510	45%	256	20%
10	5	13%	2046	8%	1024	7%
10	10	52%	2046	42%	1024	21%

Table 2: Results of 100 samples for each $n1 \times n2$ problems

Several conclusions may be drawn from these computational results. In their paper Wen and Yang [7] suggested in their conclusions that there may well be a correlation between the effectiveness of their bounding function and the ratio of the number of leader's variables, $n1$, to the number of follower's variables, $n2$. The results seem to confirm this. It is apparent from the tables that when the numbers of both leader's and follower's variables are similar both the evaluated nodes percentage and the kstar percentage figures are similar and hover around 50%. However if $n1 > n2$ then both these percentages, which measure the effectiveness of the bounding function at finding a tight upper bound for the optimal feasible solution, are significantly lower. In the case of the 10×5 problem these values drop to very low levels indicating excellent performance by both the bounding function and the algorithm in general. On the other hand it would seem that if $n2 > n1$ the effectiveness deteriorates giving the highest percentages.

Examining the performance of the algorithm and the use of the upper bounds at each level in the tree to choose branching means examining kstar. It would appear from the low values of kstar, all lower than 35%, that the addition of controlling the decision by utilizing the Z_N^U was an effective measure in tightening the bounds in the solution of the problem.

In conclusion, further work using upper bound theorems to the general bi-level integer programming problem would seem to be the most logical course to take forward from this point.

REFERENCES

[1] Bard, J.F and Falk, J.E., 1982, *An Explicit Solution to the Multi-Level Programming Problem*, Comput.Opns.Res. **9**,77-100.

[2] Bard, J.F and Falk, J.E.,1989, *The Mixed Iteger Linear Bi-level Programming Problem*,Operations Reasearch **38, No.5**, 911-921.

[3] Bard, J.F. and Moore, J.T., 1990, *A Branch and Bound Algorithm For The Bi-Level Programming Problem*,SIAM J.Sci.Stat.Comput. **Vol.11 No.2**, 281-292.

[4] Geoffrion,A.M. and Hogan,W.W, 1972, *Coordination of Two-Level Organizations with Multiple Objectives*, in *Techniques of Optimization*, A.V. Balakrishnan, Academic Press, New York.

[5] Hansen, P., Jaumard, B. and Sauvard, G., 1992, *New Branch and Bound Rules For Linear Bi-Level Programming*,SIAM J.Sci.Stat.Comput. **Vol.13 No.5**, 1194-1217.

[6] Vincente, L. Savard, G. and Judice, J., 1996, *Dicrete Linear Bilevel Programming Problem*, Journal of Optimization Theory and Applications **Vol.89 No.3**, 597-614.

[7] U.P.Wen and Y.H.Yang, 1990, *Algorithms For Solving The Mixed Integer Two-Level Linear Programming Problem*, Computters Opns. Res. **17 No.2**, 133-142.

APPENDIX

A. SAS Program Inclusion

```
%macro datdev;

    data dataLObj;
        length type $10;
        _row_="object";
        %do i=1 %to &n1;
            x&i=int(60*ranuni(0))-30;
        %end;
        %do j=1 %to &n2;
            %let c=%eval(&j+&n1);
            x&c=int(60*ranuni(0))-30;
        %end;
        type="max";
        rhs=.;
        output;
        run;

    data dataFObj;
        length type $10;
        _row_="object";
        %do i=1 %to &n1;
            x&i=0;
        %end;
    %do j=1 %to &n2;
        %let c=%eval(&j+&n1);
        x&c=int(24*ranuni(0))-12;
    %end;
    type="max";
    rhs=.;
    output;
run;


data dataFA;
    length _row_ $10 type $10;;
    %do k=1 %to &m;
        _row_="b&k";
        %do i=1 %to &n1;
            x&i=int(36*ranuni(0))-18;
        %end;
        %do j=1 %to &n2;
            %let c=%eval(&j+&n1);
            x&c=int(18*ranuni(0));
        %end;
        frac=0.25*ranuni(0)+0.5;
        type="le";
        rhs=int(frac*sum(x1 %do i=2 %to &totvar; ,x&i %end;));
        output;
    %end;
drop frac;
```

```
run;


data dataFBin;
    length type $10;
    _row_="binary";
    %do i=1 %to %eval(&n1+&n2);
        x&i=1;
    %end;
    type="binary";
    rhs=.;
    output;
run;


data dataFX;
    length _row_ $10 type $10;;
    %do k=1 %to &n1;
        _row_="x&k";
            %do i=1 %to &n1;
                x&i=0;
                x&k=1;
            %end;
            %do j=1 %to &n2;
                %let c=%eval(&j+&n1);
                x&c=0;
            %end;
        type="eq";
        rhs=0;
        output;
    %end;
run;

data dataF;

    set dataFObj dataFA dataFX dataFBin;
    _type_=type;
    _rhs_=rhs;
    drop rhs type;
run;

proc lp data=dataf primalout=Fprimal printlevel=-2;
run;

data databX;
    length _row_ $10 type $10;;
    %do k=1 %to &n2;
        %let f=%eval(&k+&n1);
        _row_="x&f";
            %do i=1 %to &n1;
                x&i=0;
            %end;
            %do j=1 %to &n2;
                %let d=%eval(&j+&n1);
                x&d=0;
                x&f=1;
                %end;
            type="le";
            rhs=1;
            output;
        %end;
    run;

    data dataB;
        set dataLObj dataFA dataBX dataFX;
        _type_=type;
        _rhs_=rhs;
        drop rhs type;
    run;

    proc lp data=dataB primalout=Bprimal dualout=Bdual printlevel=-2;
    run;

    data combo;
        set datalobj datafobj datafa;
    run;
proc print data=combo; run;
%mend datdev;



%macro iter;


proc iml;reset noflow; term=0;
```

```
    create term;

    append from term;

    close term;


start down2;


w1=0;w2=0;s1=0;s2=0;

    do k=1 to n1;

        if x[1,k]=1 then s1=s1+h[k,1];

            else if x[1,k]=. then do;

                if h[k,1] > 0  then w1=w1+h[k,1];

                end;

        if x[2,k]=1 then s2=s2+h[k,1];

            else if x[2,k]=. then do;

                if h[k,1] > 0 then w2=w2+h[k,1];

                end;

    end;

    ZU[row,5]=l;

    ZU[row,2]=ZB+s1+w1;

    ZU[row,1]=ZB+s2+w2;


    temprow=row;

    if l>=1 then do;

        if ZU[row,1]<ZU[row,2] then do;

            column=2;

            startrow=2**l;

            subrow=2*(subrow-1)+2;
            row=startrow+subrow-1;end;
        else do;
            column=1;
            startrow=2**l;
            subrow=2*(subrow-1)+1;
            row=startrow+subrow-1;end;
    end;
if temprow=2 then do;
    ZU[temprow,3]=1;
    ZU[temprow,4]=column;
    end;

if ZU[temprow,2] < ZU[temprow,1] then do;
        x[1,l]=0; ZU[temprow,6]=1;end;
            else do; x[2,l]=1;ZU[temprow,7]=1;end;
    if l < n1 then do;
        ZU[row,3]=temprow;
        ZU[row,4]=column;

        x[1,l+1]=1; x[2,l+1]=0;
        end;
    else do;
    row=temprow;
    temprow=ZU[row,3];
        create currXL from x;
        append from x;
        close currXL;
        create row from row;
        append from row;
        close row;
        create temprow from temprow;
        append from temprow;
        close temprow;

        stop;end;
finish down2;

start down; w1=0;w2=0;s1=0;s2=0;
    do k=1 to n1;
        if x[1,k]=1 then s1=s1+h[k,1];
            else if x[1,k]=. then do;
                if h[k,1] > 0  then w1=w1+h[k,1];
                end;
        if x[2,k]=1 then s2=s2+h[k,1];
            else if x[2,k]=. then do;
                if h[k,1] > 0 then w2=w2+h[k,1];
                end;
    end;

    ZU[row,5]=l;
    ZU[row,2]=ZB+s1+w1;
    ZU[row,1]=ZB+s2+w2;

    if ZU[temprow,1]<ZU[temprow,2] then do;
```

```
        column=1;end;
    else do;
        column=2;end;

ZU[row,3]=temprow;
ZU[row,4]=column;

if ZU[row,2] < ZU[row,1] then do;
    x[1,l]=0; ZU[row,6]=1;end;
        else do; x[2,l]=1;ZU[row,7]=1;end;
if l < n1 then do;
    x[1,l+1]=1; x[2,l+1]=0;
    end;
else do;
    create currXL from x;
    append from x;
    close currXL;
    create row from row;
    append from row;
    close row;
    create temprow from temprow;
    append from temprow;
    close temprow;
    stop;end;

temprow=row;
if l>=1 then do;
```

```
        if ZU[row,1]<ZU[row,2] then do;
            column=2;
            startrow=2**l;
            subrow=temprow-(2**(l-1))+1;
            row=startrow+2*(subrow-1)+column-1;

        end;
        else do;
            column=1;
            startrow=2**l;
            subrow=temprow-(2**(l-1))+1;
            row=startrow+2*(subrow-1)+column-1;

        end;
    end;
finish down;



start back;

    temprow=ZU[row,3];
    Zp=min(ZU[row,1:2]);
    if (Zp<Zstar | zu[row,6:7]={1 1}) then do;
        l=l-1;
        x[,l+1]={.,.};
        ZU[row,6:7]= {1 1};
        subrow=temprow-(2**(l-1))+1;
    row=temprow;
    return;
end;
else do;
    if l=1 & (ZU[1,6:7]={. 1} | ZU[1,6:7]={1 .}) then do;

        x=J(2,n1,.);
        if ZU[1,6:7]={. 1} then do;
        x[1:2,1:2]={0 1,0 0};
        end;
        else do;
        x[1:2,1:2]={1 1,1 0};
        end;
        ZU[row,6]=1;ZU[row,7]=1;
        l=2;row=2;run down2;
        do until((ZU[temprow,1]< Zstar &
                        ZU[temprow,2]< Zstar )| l = n1);
            w1=0;w2=0;s1=0;s2=0;
            l=l+1;
            run down2;
        end;

        end;

    if l=n1 then do;
        ZU[row,6]=1;ZU[row,7]=1;
        x[,l]=abs(x[,l]-{1,1});
            create currXL from x;

            append from x;

            close currXL;

            create row from row;

            append from row;

            close row;

            create temprow from temprow;

            append from temprow;

            close temprow;


            stop;


    end;

        else do;

            x[,l]=abs(x[,l]-{1,1});

            if ZU[row,1]< Zu[row,2] then col=1;

                else col=2;

            ZU[row,6:7]= {1 1};

            temprow=row;

            row=2*(subrow-1)+col+2**l-1;

            x[,l+1]={1,0};

            do until((ZU[temprow,1]< Zstar &

                           ZU[temprow,2]< Zstar )| l = n1);

                l=l+1;

                run down;

            end;

        end;
    end;
finish back;



use currxl;
    read all into x;
close currxl;

use l;
    read all into l;
close l;

use currZU;
    read all into ZU;
close currZU;

use ZB;
    read all into ZB;
close ZB;

use temprow;
    read all into temprow;
close temprow;

use row;
    read all into row;
close row;
```

```
use Bdual;
    read all var{_dual_} into yb where(_type_ = 'LE');
use dataFA;
    read all var{%do i=1 %to &n1;x&i %end;} into a ;

m=nrow(a); n2=nrow(yb)-nrow(a);n1=ncol(a);
    a2=J(n2,n1,0); a=a//a2;
    hp=t(yb)*a;

use datalobj;
    read all var{%do i=1 %to &n1;x&i %end;} into c1;

h=T(c1-hp);

use dataLobj;
    read all var{%do i=1 %to &totvar;x&i %end;} into cl2;

use star;
    read all var{zstar} into zstar;
close star;

do until(l=1 & ZU[1,6:7]={1 1});

    run back;

    if l=1 & ZU[1,6:7]={1 1} then do;
```

```
        term=1;

        create term from term;

        append from term;

        close term;

    end;


end;


create currZU from ZU;

    append from ZU;

close currZU; create temprow from temprow;

    append from temprow;

close temprow;


quit;




data _null_;

    set term;

    call symput('term',col1);

run;

%if &term ne 1 %then %lprob;


%mend iter;




%macro Lprob;
```

```
%let lpcount=%eval(&lpcount+1);
proc transpose data=currxl out=xltrans;run;


data rhs;
    set xltrans; keep col1; rename col1= rhs;
run;


data datalx;
    set datafx;drop rhs;
run;


data datalx2;
    merge datalx rhs;
run;


data dataL;


set dataFObj dataFA datalx2 dataFBin;
         _type_=type; _rhs_=rhs; drop type rhs;
run;


proc lp data=dataL primalout=Lprimal printlevel=-2; run;
%let sc1=%scan(&_orlp_,1);
%let sc2=%scan(&sc1,2,=);
proc iml;
    sc2="&sc2";
```

```
use dataLobj;
read all var{%do i=1 %to &totvar;x&i %end;} into cl2;
close dataLobj;
use Lprimal;
read all var{_value_} into xvl where(_type_ ='BINARY');
close Lprimal;
xvlt=T(xvl);varnames={'Zstar'};
use star;
read var{zstar} into Zstar;
close star;
ZL=cl2*xvl;
use currZU;
read all into ZUstar1;
close currZU;
if sc2 ^= "INFEASIBLE" then do;
    if ZL>Zstar then do;
        Zstar=ZL;
        star=zstar||xvlt;
        create star from star [colname=varnames];
        append from star;
        close star;
        create ZUstar from ZUstar1;
        append from ZUstar1;
        close ZUstar;
    end;
else do;
    end;
    end;
quit; data star;
    set star;
        %do j=1 %to &totvar;
        %let f=%eval(&j+1);
        rename col&f=x&j;
        %end;
run;

%mend lprob;


%macro begin;
%datdev;
%let totvar=%eval(&n1+&n2);
proc iml; start down; w1=0;w2=0;s1=0;s2=0;
    do k=1 to n1;
        if x[1,k]=1 then s1=s1+h[k,1];
            else if x[1,k]=. then do;
                if h[k,1] > 0  then w1=w1+h[k,1];
                end;
        if x[2,k]=1 then s2=s2+h[k,1];
            else if x[2,k]=. then do;
                if h[k,1] > 0 then w2=w2+h[k,1];
                end;
    end;
    ZU[row,5]=1;
    ZU[row,2]=ZB+s1+w1;

    ZU[row,1]=ZB+s2+w2;


    temprow=row;

    if l>=1 then do;

        if ZU[row,1]<ZU[row,2] then do;

            column=2;

            startrow=2**l;

            subrow=2*(subrow-1)+2;

            row=startrow+subrow-1;end;

        else do;

            column=1;

            startrow=2**l;

            subrow=2*(subrow-1)+1;

            row=startrow+subrow-1;end;

    end;

if temprow=1 then do;

    ZU[temprow,3]=1;

    ZU[temprow,4]=column;

    end;


if ZU[temprow,2] < ZU[temprow,1] then do;

        x[1,l]=0; ZU[temprow,6]=1;end;

            else do; x[2,l]=1;ZU[temprow,7]=1;end;

    if l < n1 then do;

        ZU[row,3]=temprow;

        ZU[row,4]=column;
        x[1,l+1]=1; x[2,l+1]=0;
        end;
finish down;


    use Bdual;
    read all var{_dual_} into yb where(_type_ = 'LE');
    use dataFA;
    read all var{%do i=1 %to &n1;x&i %end;} into a ;
    m=nrow(a); n2=nrow(yb)-nrow(a);n1=ncol(a);
    a2=J(n2,n1,0); a=a//a2;
    hp=t(yb)*a;
    use datalobj;
    read all var{%do i=1 %to &n1;x&i %end;} into c1;
    h=T(c1-hp);


    use dataLobj;
    read all var{%do i=1 %to &totvar;x&i %end;} into cl2;


    use Bprimal;
    read all var{_value_} into xvB where(_type_ ='NON-NEG');


    ZB=cl2*xvB;
    create ZB from ZB;
    append from ZB;
    close ZB;
    use Fprimal;
    read all var{_value_} into xvF where(_type_ ='BINARY');
    xvFt=T(xvF);varnames={'Zstar'};

    Zstar=cl2*xvF;

    star=zstar||xvft;
    create star from star [colname=varnames];
    append from star;
    close star;

    ZU=J(2**n1-1,7,.);
    x=J(2,n1,.);
    x[,1]={1,0};

row=1;subrow=1;l=1;temprow=1; run down;
do until((ZU[temprow,1]< Zstar & ZU[temprow,2]< Zstar )| l = n1);
w1=0;w2=0;s1=0;s2=0; l=l+1; run down;

end; row=temprow; temprow=ZU[row,3];

create currXL from x;
append from x;
close currXL;
create currZU from ZU;
append from ZU;
close currZU;
```

```
create temprow from temprow; append from temprow;

close temprow; create row from row; append from row; closerow;

create l from l; append from l; close l;


quit; data star;

    set star;

        %do j=1 %to &totvar;

        %let f=%eval(&j+1);

        rename col&f=x&j;

        %end;

run;


%mend begin;




%macro backtrack;

%do %until(&term=1);


%iter;


data _null_;

    set term;

    call symput('term',col1);

run;


%end;

%let time2=%sysfunc(time());
```

```
%let time=%sysevalf(&time2-&time1);

%let minute=%sysfunc(minute(&time));

%let second=%sysfunc(second(&time),5.2);

%let leaves=%eval(2**(&n1);

%put Total processing time is &minute minutes, &second seconds;

ods listing close; proc means data=zustar n nmiss;

    var col1 col2;

    ods output summary=stuffa;

run; data stuffa;

    set stuffa;

    kstar=(col1_N + col2_N);

    keep kstar;

run; proc means data=currzu n nmiss;

    var col1 col2;

    ods output summary=stuffb;

run;


data stuffb;

    set stuffb;

    n1=input(symget('n1'),5.);

    n2=input(symget('n2'),5.);

    m=input(symget('m'),5.);

    i=input(symget('nt'),5.);

    pct=(col1_N+col2_N)/(col1_N+col2_N+col1_Nmiss+col2_Nmiss);

    evaluated_nodes=(col1_N+col2_N);

    total_nodes=(col1_N+col2_N+col1_Nmiss+col2_Nmiss);

    lpcalls=input(symget('lpcount'),10.);
    leaves=input(symget('leaves'),10.);
    Minutes=input(symget('minute'),10.2);
    Seconds=input(symget('second'),10.2);

    keep n1 n2 m i evaluated_nodes
              total_nodes lpcalls leaves minutes seconds;
run;
data stuff&nt;
    set stuffb stuffa;
    merge stuffb stuffa;
run;
ods listing;
proc print data=star;
run;
proc print data=stuff&nt;
run;

%mend backtrack;
%macro dotimes;
options nonotes nosource dkrocond=nowarn;

%do nt=1 %to &nr;
%let time1=%sysfunc(time());
%let lpcount=0;
%let totvar=%eval(&n1+&n2);

%begin;
```

```
%lprob;


%backtrack;

%end;

data allstuff;

    set

        %do j=1 %to &nr;

        stuff&j

        %end;;

run;


proc export data= work.allstuff

  outfile= "f:\Thesis stuff-Rem\Thesis Final\output&&&n1&n2..xls"

  dbms=excel2000 replace;

run;

%mend dotimes;
```

Then running the following control sequence
 will execute the above macros:

```
%let nr = # of repetitions required;

%let n1 = # of leader's variables;

%let n2 = # of follower's variables;

%let m = # of constraints;

%dotimes;\\
```