(2)

AD-A236 023

ENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 25 Feb 91	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE An Algorithm for Solving a Class of Pseudo-Boolean Equations		5. FUNDING NUMBERS Master's Thesis
6. AUTHOR(S) Kevin J. Loy, CPT, U.S. Army		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Colorado School of Mines Department of Mathematics and Computer Sciences Golden, CO 80401		8. PERFORMING ORGANIZATION REPORT NUMBER T-4003
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQDA, MILPERCEN (DAPC-OPB-D)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
This is a thesis submitted to the Faculty and Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirement for the degree Master of Science (Mathematics).

12a. DISTRIBUTION/AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A new algorithm named the "Overall Algorithm" applicable to a 0-1 variable (Pseudo-Boolean) system of equations with an objective function and a set of inequality constraints. The objective equation can be linear or non-linear. The set of constraints must be either all linear or all non-linear independent of the objective function. The Overall Algorithm first solves the objective function and checks the set of constraints for feasibility. If the solution is feasible then the algorithm stops. If the solution is not feasible then the Overall Algorithm moves to the set of constraints and solves the set of constraints for a feasible solution. If there is a solution to the constraints then the Overall Algorithm creates bounds for the optimal solution. This is conjectured through application to 48 previously published problems that are listed in the thesis. The Overall Algorithm is significant because it is an alternative method for the solution of Pseudo-Boolean systems of equations or models. It is a simple algorithm that, based on computational experience, has been shown to be surprisingly accurate. Finally, the most iterations required for this algorithm is n, where n is the number of variables.

14. SUBJECT TERMS Algorithm, 0-1 Integer Programming, Capital Budgeting, Decision Theory, Pseudo-Boolean, Unconstrained 0-1 Maximization/Minimization			15. NUMBER OF PAGES 121
			16. PRICE CODE
unclas	unclas	unclas	
17. SECURITY CLASSIFICATION OF REPORT unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT unclassified	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298, 880922 Draft
Prescribed by ANSI Std. 239-18

T-4003

AN ALGORITHM FOR SOLVING A CLASS OF PSEUDO-BOOLEAN EQUATIONS

by

Kevin J. Loy

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Masters of Science (Mathematics).

Golden, Colorado

Date 2/25/91

Signed: [illegible]
Kevin J. Loy

Approved: [illegible]
Dr. R.E.D. Woolsey
Thesis Advisor

Golden, Colorado
Date 2/25/91

[illegible]
Dr. Ardel J. Boes
Professor and Head
Department of Mathematics

ABSTRACT

A new algorithm which was named the "Overall Algorithm" is applicable to a 0-1 variable (Pseudo-Boolean) system of equations with an objective equation, or function, and a set of inequality constraints. The objective equation can be linear or non-linear. The set of constraints must be either all linear or all non-linear independent of the objective equation.

The Overall Algorithm first solves the objective equation and checks the set of constraints for feasibility. If the solution is feasible then the algorithm stops. If the solution is not feasible then the Overall Algorithm moves to the set of constraints and solves the set of constraints for a feasible solution. If there is a solution to the constraints then the Overall Algorithm creates bounds for the optimal solution. This is conjectured through application to 48 previously published problems that are listed in the thesis. In addition to 48 problems there is a computer code in C-language for that portion of the algorithm that deals with the objective equation.

This Overall Algorithm is significant because it is an alternative method for the solution of Pseudo-Boolean systems of equations or models. It is a simple algorithm that, based on computational experience, has been shown to be surprisingly accurate. Finally, the most iterations required for this algorithm is n, where n is the number of variables.

Table of Contents

Table of Figures and Tables

ACKNOWLEDGEMENTS

First and foremost I would like to thank God for the support of my friends and family in completing this thesis. Without his guidance and strength it would have been impossible to accomplish the mission.

Second I would like to recognize the support and patience of my wife, Maureen. William, James, Mathew, and Mary are without a doubt a very difficult group of children to raise by yourself. Although at times thin, her patience with the children allowed me the time to produce this thesis. I love you.

Third is Dr. Woolsey. He has the unique gift of analyzing an extremely difficult math problem and understand what is happening in the problem. He can then explain the problem in plain english which is the sign of true intelligence.

Next is my committee and the Math department faculty. Dr. Maurer was able to define a new supremum in grammar excellence and her advice in mathematics was also invaluable. She is an asset that should be a must for all graduate students. Also, I would like to thank Dr. Underwood who, in my opinion, has the gift of mathematical vision. His insight in understanding my algorithm gave me a new perspective on the algorithm.

Finally, are my peers and friends. Since they are too numerous to mention I will only name a few. Mark Tillman who, on the day of my defense, in a valorous attempt to break the tension hid my suit. Others are James Watson who provided expert C-language advice, Bob Clayton who provided advice on the algorithm, and Joe Huber who provided advice on the algorithm and the defense.

I may have been verbose in my expression of thanks but, that only further demonstrates my sincere thanks to those who deserve it.

Chapter 1
BIBLIOGRAPHY AND BACKGROUND

1.1 Introduction.

The research the author conducted with respect to previous work done in this area indicated that before 1960 there were was little done in the area of Pseudo-Boolean Programming or the solving of these types of equations. G.B. Dantzig first pointed out the real importance of bivalent (0-1) variables in 1957 (Hammer and Rudeanu, 1968). From 1960 until now there are few people that have developed algorithms to solve the type of problem discussed. The following is a brief discussion of some of those individuals and their research. The research conducted currently indicates that there is no algorithm which is similar to the algorithm described in this thesis.

1.2 Problem Description.

The Overall Algorithm developed here is applicable to a class of problems that are defined, in general, in Chapter 2 with examples in Appendix A. The function, which the author will call the objective equation, maps the elements 0 and 1 of the domain into the range of integer numbers. As a result, all variables that define the objective equation or constraints are either 0 or 1. The objective equation is a linear combination of terms that are either one variable or cross products of variables. In this thesis this objective equation can be described as $f(x_1, x_2, \ldots, x_n)$. An example of an unconstrained, non-linear objective equation is:

$$\text{Min f } = 2x_1 + 3x_2 - 7x_3 - 5x_1x_2x_3 + 3x_2x_4 + 9x_4x_5 \tag{1.1}$$

The objective equation can either be unconstrained or constrained. If constrained, then the constraints must either be all linear or all non-linear. An example of a constrained objective equation where all the constraints are linear is:

$$\begin{aligned}
&\text{Minimize } Z = 6x_1 + 3x_2 + 9x_3 + 6x_4 + 12x_5 \\
&\text{s.t.} \\
&3x_1 + 6x_2 + 3x_3 + 3x_4 + 6x_5 \geq 12 \\
&21x_1 + 3x_2 \qquad - 9x_4 + 9x_5 \geq 6 \\
&-9x_1 + 9x_2 + 6x_3 \qquad - 3x_5 \geq -3
\end{aligned}$$

An example of a constrained objective equation where all the constraints are non-linear is:

$$\begin{aligned}
&\text{Maximize } 5x_1x_3 - 2x_1x_3x_4 + 4x_2x_3x_4 - 3x_3x_4 + 4x_2x_4 \\
&\text{s.t.} \\
&6x_3x_4 + x_1x_3x_4 + 3x_1x_2 \leq 6 \\
&-2x_2x_3 + 5x_1x_4 - 2x_1x_2x_4 + 5x_3x_4 \leq 5 \\
&x_1x_2 + 2x_2x_3 - x_2 - x_1x_2x_3 + 2x_3x_4 \leq 2
\end{aligned}$$

In either case, the constraints are all linear or all non-linear although the terms could be mixed. In the linear case all the terms must be linear or just contain one variable. But in the non-linear case, there may be terms that contain just one variable but because it is part of a constraint that contains terms with cross products the constraint is non-linear.

1.3 Egon Balas.

Egon Balas has conducted much research in the Pseudo-Boolean area as well as other areas. In the area of Pseudo-Boolean Programming and problem solving he has developed, and proven, an algorithm which basically uses combinatorial method of solving a system of linear equations (Balas, 1965).

> The method is essentially a tree-search algorithm that uses information generated in the search to exclude portions of the tree from consideration. (Glover and Zionts, 1965, 546)

1.4 Peter L. Hammer and Sergiu Rudeanu.

Dr. Hammer and Dr. Rudeanu described a method of solving Pseudo-Boolean equations in 1967. The basic idea of the method is the concept of a "Characteristic function". They start with a dynamic technique for minimizing an unrestricted equation and then incorporate a branching technique based on a set of rules. Combining these two techniques with the concept of a "Characteristic function" they solve unrestricted and restricted, linear and non-linear Pseudo-Boolean equations.

1.5 E. L. Lawler and M. D. Bell.

Hamdy A. Taha (Taha, 1975) describes Lawler and Bell's method that uses explicit enumeration to solve a non-linear Pseudo-Boolean set of equations. The algorithm groups the terms in the objective equation together. The algorithm also groups terms together in the constraints but it then solves the set of equations by applying a set of rules to both the objective equation and the set of constraints. The algorithm uses binary addition to skip over a solution set that would not be feasible and thereby reduce the number of iterations required to solve the problem. There is one important requirement to this algorithm as Hamdy A. Taha states:

> An important restriction on the above problem is that each of the functions $g_{01}, g_{02}, g_{11}, g_{12}, \ldots, g_{m1}, g_{m2}$ is monotone non-decreasing in each of the variables $x_1, x_2, \ldots, x_n$.(Taha, 1975)

1.6 F. Glover.

Hamdy A. Taha describes in his book, Integer Programming, 1975 Glover's method of "Enumeration Scheme." The method starts with a partial solution then enumerates through the set of variables. If the algorithm encounters a solution that is not feasible then it eliminates that partial solution without actually being considered.

1.7 Research on Algorithm Development.

The algorithm took several shapes during its evolution to its current state. As the algorithm was developed there were problems that were needed to test the algorithm. In Appendix A are some of the problems that the author used to demonstrate the capabilities of the algorithm. The algorithm originally started as just Algorithm 1. Later in the research, Algorithm 2 was developed to solve the system of constraints because the author felt it necessary to address the constraints. Hence, the Overall Algorithm was developed to tie the two algorithms together.

1.8 Conclusion.

The author conducted a thorough research to ensure the originality of the algorithm developed in this thesis. Although it does use a type of enumeration, the algorithm is not similar to any of the before mentioned algorithms that use enumeration. Also, although the algorithm does group the negative terms and the positive terms together like the method of Lawler and Bell, it does not create a set of functions. Finally, all of the above mentioned attempt to solve a system of equations considering both the objective equation and the constraints simultaneously. The algorithm that will be described in detail in the next chapter does not consider both at the same time.

Chapter 2
PROBLEM DEFINITION AND SOLUTION

2.1 Problem Definition.

The functions f and g_i are either linear or non-linear. As defined in Chapter 1, all of the g_i are the same with respect to linearity. The inequalities can be mixed $\le$ or $\ge$ within the set of constraints. The following are the general forms of the class of problems for which the algorithm is applicable.

2.1.1 Minimize form.

Consider the following problem:

Minimize

$$z = f(x_1, x_2, x_3, \dots, x_n)$$

subject to:

$$g_i(x_1, x_2, \dots, x_n) \ge b_i \qquad i \in \{1, 2, \dots, m\}$$

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

$$m, n \le 10$$

2.1.2 Maximize form.

The problem definition can also be in the form:

Maximize

$$z = f(x_1, x_2, x_3, \dots, x_n)$$

subject to:

$$g_i(x_1, x_2, \dots, x_n) \leq b_i \qquad i \in \{1, 2, \dots, m\}$$

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

$$m, n \leq 10$$

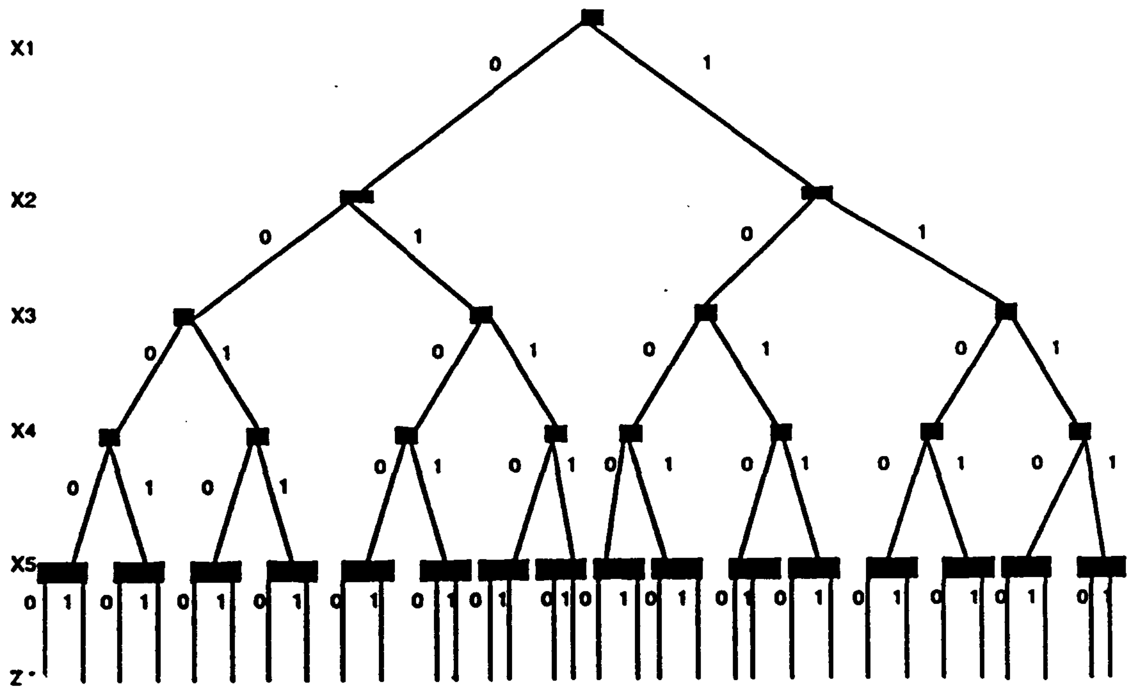
2.1.3 Practical Applicability.

Pseudo-Boolean Programming has several applications in the area of mathematics. The first is in combinatorial operations research (Hammer and Rudenau, 1967), the second could be reliability testing for a system and the third could be in electric circuits. Another area as stated by Egon Balas is economics:

> It is well known that important classes of economic (and not only economic) problems find their mathematical models in linear programs with integer variables. Prominent among these problems are those that correspond to linear programs with variables taking only one of the values 0 or 1. (Balas,1964)

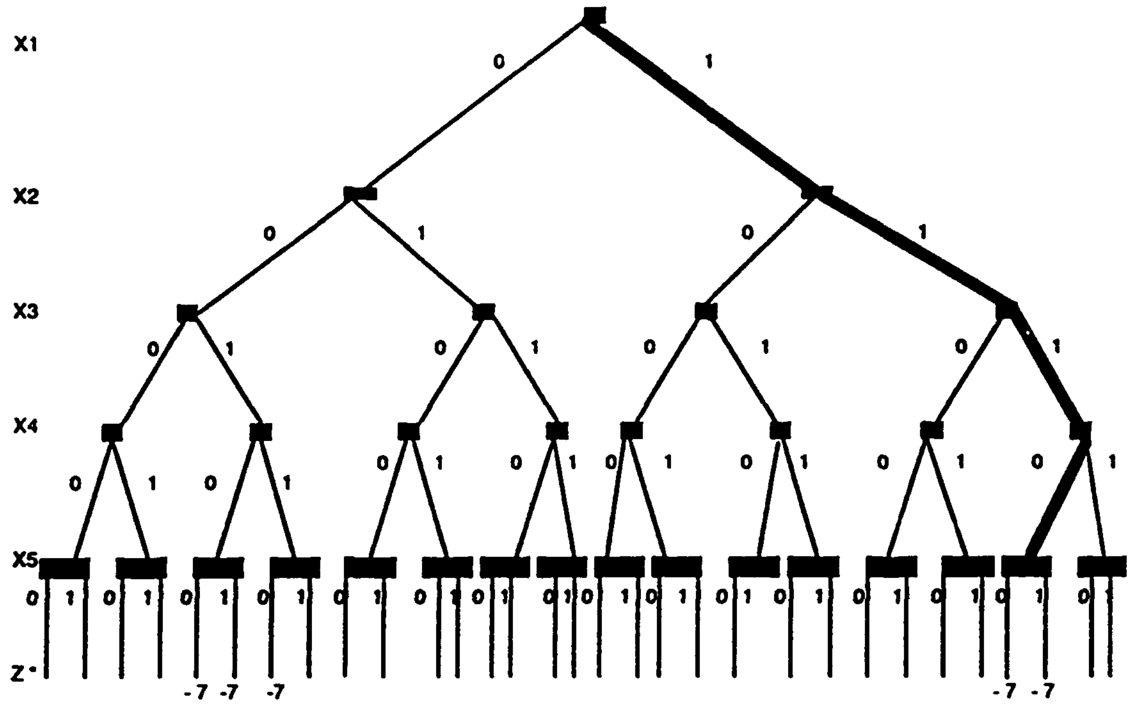
2.2 Proposed Solution.

As discussed in Chapter 1 there are currently several algorithms developed and proven to solve this type of problem. The algorithm in this thesis uses several concepts that are different from those discussed in Chapter 1. Before discussed it is appropriate to explain the solution that is generated. The overall algorithm has been shown to provide one solution to the system of equations, if it exists. Algorithm 1 provides one solution to the unconstrained objective equation. This is one disadvantage of the major concept behind the algorithm. This concept is the ratio and ranking that is done by the algorithm. There are essentially two ratios generated, the term ratio and the variable ratio. The term ratios are used to generate the variable ratios. The variable ratio is a number and the ranking uses this number to rank the variable in increasing order. The ranking puts the least desirable variable first and the most desirable variable last. Least desirable is defined as every coefficient of a least desirable variable is positive in a minimization equation. Most desirable is defined as every coefficient of a most desirable variable is negative in a minimize equation. This was designed, basically, to facilitate the user to move toward the best solution, according to the algorithm. The ranking also defines the branch of the solution the algorithm will follow. For example, the Overall algorithm generates a ranking of variables 4, 5, 2, 1, 3 for eq. 1.1. We start with the general binary tree as shown in figure 2.1 which identifies each variable and the possible value of each variable. At the end of each branch of the tree is a possible solution to the eq. 1.1. The problem is to find the combination of branches that will provide the best solution to the equation.



2.1 Binary Tree 1

All the variables are initially set equal to 1. The algorithm will iterate through the rank order established, changing the first variable in the rank order to zero and deriving a solution to the equation. The algorithm will compare each solution to the previous one every time it changes a variable to zero. Since the algorithm is always applied to minimization equation it will iterate through the rank order until a minimum is found. Therefore, once the previous solution is less than the current solution the algorithm stops. In our example, seen is section 1.2, it found the solution of z* = -7 and solution set (1,1,1,0,0). Figure 2.2 shows one path to the solution z* = -7. Also identified on figure 2.2 are the other solution sets that obtain the solution z* = -7. They are identified by the solution at the end of the branch.



2.2 Binary Tree 2

The following paragraphs describe the reason for the various steps listed in order of execution rather than importance to facilitate the reader in comparing them to the steps listed in Chapter 3.

First, the overall algorithm uses Algorithm 1 to derive a solution to the objective equation. It converts the problem to a minimization problem. This is important because the algorithm uses two ratios that require the problem to be a minimization problem.

Second, a ratio for each term is created with the coefficient of the term as the numerator and the number of variables in the term as the denominator. This is the contribution that each variable in the term contributes toward the solution. It also could be the amount of impact that coefficient will have on the solution. As one can see, the more variables in the term the less likely that the term will impact on the solution because there is a higher probability that the term will be zero, unless all of the variables in that term are equal to one.

Each time a variable is present in a term the term ratio is added to a positive group if the coefficient is positive and a negative group if the coefficient is negative. These groups of ratios are summed and a second ratio is created. This second ratio is the effect the variable has on the solution. The ratio is the negative sum over the positive sum. If there are only negative terms that contain the variable in question then it is most advantageous to retain that variable in the solution set. Therefore, the notation of infinity is given to that ratio and is ranked highest. Likewise, if there are only positive terms that contain the variable in question then it is a disadvantageous to retain that variable. Therefore, the value of zero is given to that ratio and is ranked the lowest.

At this stage there are a few points worth noting. Some variables may have ratios that are equal to other ratios. If the equality is among two or more variables that have ratios equal to zero then it is unimportant because these variables will be zero in the final solution. If the equality is among two or more variables that have ratios equal to infinity then this again is unimportant because these variables will be one in the final solution. The interesting equality is among two or more variables where the ratios are between zero and infinity. When there is a tie, as demonstrated in Chapter 4, the algorithm refers to the sum of the positive term ratios to break the tie. The variables are ranked in decreasing order because they are considered the least favorable according to the impact they have on the solution in terms of adding a positive coefficient. If there is still a tie then this appears to indicate that the variables are of equal importance. This is an area of possible solution sets rather than a solution. Finally, the ratio of the negative value over the positive value is why the problem is converted to a minimization problem. As explained the ratios are ranked in increasing order and the variables are changed to zero in the rank order until a minimum is found. Also, the constraints are converted from ≤ to ≥ as a matter of convention and because the ratios for the constraints, like the ratios for the objective equation, are derived to find a minimum. The author believes the algorithm could be changed to a maximize and the constraints converted from ≥ to ≤ if someone were to:

1. Create ratios of positive over negative and,
2. Change the variables from one to zero in the rank order established until a maximum is found.

Once Algorithm 1 derives a solution the Overall Algorithm applies this solution to the system of constraints, if any. If the constraints are satisfied then the Overall Algorithm stops. If they are not satisfied then the Overall Algorithm moves to Algorithm 2. Algorithm 2 derives a solution set, if it exists, in much the same manner as Algorithm 1. The only exception is the use of the y variable. Algorithm 2 then uses the objective equation to derive a solution. It returns to the Overall Algorithm and the Overall Algorithm uses the solution from Algorithm 2 and 1 to establish a lower and upper bound for the optimum solution.

2.3 Code of Algorithm 1.

As a part of the proposed solution the author has coded Algorithm 1 in C computer code (Appendix B). This algorithm deals with the objective equation and as a result will provide a lower bound for the solution, if the constraints are not satisfied. The code only uses the four basic math functions and a sort function (Press et al, 1988). The code currently will accept 30 variables and 30 terms but that can be changed by changing the declaration of variables ROW and COL at the beginning of the code. The data is stored in a way that makes the computations of the different ratios very simple. For example, consider the equation: Minimize $z = 2x_1 - 3x_1x_3 + 4x_2x_3$. The data for this problem will be stored in an array as shown in figure 2.3 below. Therefore, the coefficient for the second term is stored in **terms(2,0) and the only variable is stored in **terms(2,3).

		coefficient	x_1	x_2	x_3
**terms	term 1	2	1	0	0
array in	term 2	-3	0	0	1
C-code	term 3	4	0	1	1

2.3 Array of data for C code

It is now a very simple process to sum the ones in the rows to obtain the number of variables in a particular term. Also, it is easy to create the term ratio by dividing the coefficient, in column zero, by the sum of the variables in that term. Finally, when scanning the terms for each variable, if a variable is present then there will be a one in the variable column.

2.4 Pseudo - code of Algorithm 1.

The previous computer code can be condensed into nine mathematical steps called a pseudo - code which is listed in Appendix G. The objective equation is defined in mathematical terms but, it is the same as defined at the beginning of this chapter.

Chapter 3
METHODOLOGY

3.1 Introduction.

The Overall Algorithm is described in detail is the following pages. After each step is a brief explanation of the logic behind the step. The Overall Algorithm ties Algorithm 1 and Algorithm 2 together. The flowchart of each algorithm is contained in Appendix C and the worksheets are contained in Appendix D. Algorithm 1 is coded in C-language and the code is contained in Appendix B. Finally, the condensed word version of the algorithm is contained in Appendix E.

3.2 Overall Algorithm.

This algorithm is designed for solving the class of problems described in Chapters 1 and 2. The class of problems usually consists of an objective equation and one or more constraints. The constraints should be inequalities. This algorithm essentially uses the solution set from Algorithm 1 for the objective equation and applies it to the set of constraints. If they are satisfied then it stops. If the constraints are not satisfied then the last variable in the rank order established in Algorithm 1 is changed to one and the new solution set is applied to the set of constraints. The algorithm is basically moving backwards on the binary tree to find a solution to the system of equations. If the constraints are not satisfied then this algorithm moves to Algorithm 2 and attempts to satisfy the set of constraints. The author does not claim optimality.

3.2.1 Step 1

Solve for the minimum value and the solution set of the Objective Equation (OE) using Algorithm 1. Take the solution set of the OE and apply it to the System of Constraints (SC).

3.2.2 Step 2

Are the SC satisfied? If yes then end the algorithm because the solution set is feasible. If not then go to Step 3.

Steps 1 and 2 are done because the OE may not be restricted by the constraints.

3.2.3 Step 3

Are there any zero variables in the solution set? If yes then change the last zero variable of the rank order established in Algorithm 1 of the solution set to one, apply this solution set to the SC, and return to Step 2. If no then go to Step 4.

As stated earlier, the variables are ranked in order of least desirable to most desirable. By changing the last zero variable in the rank order from zero to one we are adding the next variable that would increase the minimum value the least. Algorithm 2, for the constraints, has been shown to not be optimal. Therefore, if a feasible solution can be derived without Algorithm 2 then this will save time because the user will not be required to use Algorithm 2.

3.2.4 Step 4

Solve for the minimum value and solution set of the SC using Algorithm 2. If the solution set exists the go to Step 5. If it does not exist then stop algorithm because solution does not exist.

In the event Algorithm 1 does not provide a solution set then we must employ Algorithm 2 and focus on the solution set for the constraints.

3.2.5 Step 5

At this point the algorithm can only bound the actual value of the OE. Using the minimum value from Step 1 and the minimum value from Step 4 create a bound with a lower bound from Step 1 and an upper bound from Step 4. End the algorithm.

Although Algorithm 2 does not always provide the optimal solution set it has been shown to provide a feasible solution set when it exists. This solution set will provide a solution from the OE and is usually different from the solution derived in Algorithm 1. Hence, the interval is created although it is possible for both algorithms to derive the same solution and different solution sets.

3.3 Algorithm 1 for the Objective Equation.

This algorithm is applicable to all 0,1 variable unconstrained or constrained problems. It is a sub algorithm or step of the Overall Algorithm. Specifically, it is step 1 of the Overall Algorithm. This algorithm solves the objective equation and derives a solution set (x*) and solution (z*). Use the worksheet provided in Appendix D to record information. The number of each step corresponds to the number of the block on the worksheet. Also, at Appendix C is the code for this algorithm.

3.3.1 Step 1

Is the equation a minimize equation? If no then multiply the problem by -1. Record the minimum equation in block 1

This step is done because when the algorithm iterates through the objective equation it initially sets all the variables equal to one. Each time it iterates a variable in the rank order is set equal to zero. It derives a new solution and compares the previous solution to the new solution. The comparison is which solution is less. The algorithm iterates until a minimum is found.

3.3.2 Step 2

Multiply the objective equation out. Example:

Minimize

$$-2x_1(1-x_2)+x_3(1-x_4)=-2x_1+2x_1x_2+x_3-x_3x_4$$

Record this step in block 2.

The algorithm uses the coefficient of each term and divides the coefficient by the number of variables in that term. Therefore, its relies on the fact that the objective equation is in its reduced form. Also, the algorithm can only solve for the actual value of each variable. For example, it can solve for x_1 but not $(1- x_1)$. Therefore, each variable must be in the objective equation as its actual form.

3.3.3 Step 3

Record the positive terms of the equation in the block marked POS. Record the **absolute** value of the negative terms in the block marked NEG. In the block marked RATIO record the ratio of the absolute value of the coefficient of each term over the number of variables in that term.

Step 3a

Record each variable in column 3a putting one variable in each row.

This step really has no mathematical or logical reasoning behind it. It is done for to facilitate the user of the algorithm in organizing the information.

3.3.4 Step 4

Scan each term in the positive block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio below the term in column 4 for the first variable.

NOTE: If the first variable is not present in any of the positive terms then record 0 for that variable in column 4.

Step 4a

Sum the recorded answers in column 4 for the first variable and record that value in column 4a of that variable row.

The ratio in step 3 is the term ratio. It is considered, by the author, to be the contribution of each variable in that term to the objective equation. This contribution may be positive (unfavorable) or negative (favorable). In this case it is the positive and when all of these contributions are summed they become the total positive contribution to the objective equation by that particular variable. In a sense, this step also ties the variables in each term together because if there are allot of variables in a term then this ratio will be small. Likewise, if there are very few variables then this ratio will be larger relative to having allot of variables in the term.

3.3.5 Step 5

Repeat step 4 and 4a for each variable in column 3a. After this step you should have a real number recorded in column 4a for each variable in column 3a.

Step 4 was an example looking at one variable. Step 5 is just a continuation of step 4 to the rest of the variables that are in column 3a.

3.3.6 Step 6

Scan each term in the negative block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio below the term in column 6 of that variable row.

NOTE: If the first variable is not present in any of the negative terms then record 0 for that variable in column 6.

Step 6a

Sum the recorded answers in column 6 for the first variable and record that value in column 6a of that variable row.

This step is the same process as step 4 but for the negative term ratios.

3.3.7 Step 7

Repeat step 6 and 6a for each variable in column 3a. After this step you should have a real number recorded in column 6a for each variable in column 3a.

3.3.8 Step 8

Create a ratio for each variable of the number recorded in column 6a over the number recorded in column 4a. Is there a zero in column 4a? If yes, then record infinity (i.e., ∞) for the ratio of that variable in column 8. Do this for all variables that have a zero in column 4a.

This step creates the variable ratio that was mentioned earlier. This step, the first step, and step 9 are all tied together. As one can see if the ratio is large then that variable contributes favorably to the objective equation because there is more negative contribution (numerator) relative to the positive contribution (denominator). As a result it is tied to the first step because the objective equation is always a minimization function and the ranking in step 9 puts the variables in increasing order according to these variable ratios. Therefore, the variable with the most favorable contribution will be ranked highest and changed from one to zero last in the iteration portion of the algorithm.

3.3.9 Step 9

In increasing order, rank each variable according to the ratio created in column 8. The variable with the smallest ratio is in column 1 of block 9, etc. Record one for the value of each variable in the value row of block 9.NOTE 1: If more than one ratio is equal to zero, then rank those variables first among the other variables. Then rank them according to the decreasing values from column 4a. If more than one ratio equal to infinity then rank them last among the other variables. Then rank them according the increasing value from column 6a.

NOTE 2: If there are ties of the non-zero and non-infinity ratios in column 8 then refer to column 4a and rank them in decreasing order.

NOTE 3: If there are ties in either 6a or 4a then break them arbitrarily.

As mentioned earlier, this step is tied to the first step because this step ranks the variables according to their total contribution. Based on how the variable ratio is constructed it makes sense that the closer to zero the ratio is the more unfavorable that variable will be for the objective equation. Note 1 (i.e., the variable ratio is zero) puts those variables that have no favorable contribution to the objective equation first in the rank order and consequently they are changed to zero first. The reason they are ranked in decreasing order among their set is because intuitively you want the variable that contributes the most positive value to the objective equation changed to zero first, therefore, it is ranked first. Again, the same logic is applied to note 2. Column 4a is the positive contribution of the variable to the objective equation and hence you want those variables with the most positive contribution to the objective equation changed to zero first for this group of variables. If the variables are equal at this point then there is no other criteria to separate the variables and the variables in question probably have the same contribution to the objective equation. In other words, the variables that are tied have the same variable ratio, the same positive contribution and consequently the same negative contribution. Hence, the variables in question probably have the same contribution to the objective equation.

3.3.10 Step 10

Using the equation from block 2 solve for a z* value where all variables are equal to 1. Call this value z* old and record z* old in block 10. Record the current value of the variables in block 10 as x* old in vector form (i.e., (0,1,1,1)) in the rank order established in Step 9.

This step establishes the initial z* value to start the iterations and comparisons of z* new with z* old. The x* is recorded with the z* as a matter of record keeping so the user does not get confused as to which z* was derived from which x*.

3.3.11 Step 11

Cross out any previous z* new and x* new in block 11. Change the first non-zero variable in the rank order in block 9 to 0 by crossing out the 1 below the variable in block 9 and record 0. Obtain a new z* value, with the new variable values, from the equation in block 2. Call this value z* new and record z* new in block 11. Call the new variable values x* new. Record x* new as a vector (i.e., (0,1,1,1)) in the rank order established in Step 9. Record x* new in the corresponding z* new row of block 11.

This step changes a variable from one to zero and obtains a new z* value. Each time this step is done it eliminates a variable that has more unfavorable contribution to the objective equation. It will change variables to zero until a variable is changed to zero that has more favorable contribution than unfavorable. It is important to note that one can not look at the variable ratios and decide that all ratios less than a value, say one, contribute favorable and should be equal to one and all the other variables are equal to zero. This cannot be done because all of the ratios can be less than one or all greater than one. The point to be made is that there is no fixed number that can be used as decision criteria.

3.3.12 Step 12

If the z* old is less than the z* new then stop and go to step 14. Else, cross out z* old and x* old in block 10 and record z* new and x* new in block 10 as z* old and x* old. Go to step 13.

Since z* old is less than z* new this means that a variable that has favorable contribution was changed to zero. It is a strictly less than because this will force all the variables to zero as possible without increasing the z*. If the z* remains unchanged for more than one iteration then the variable that was changed to zero during the iteration could either be zero or one. The author has also found that the z* that did not change during the iteration is usually the final z* value.

3.3.13 Step 13

Is there a non-zero variable remaining in block 9? If yes, then repeat step 11 for the next non-zero variable. If no, then go to Step 14.

Step 13 is merely a way to end the algorithm if all of the variables have been changed to zero. An example of when this can happen is when all of the coefficients are positive.

3.3.14 Step 14

The minimum value of the equation is z* old in block 10. The current values of the variables in block 10, the x* old column, is the solution set. Match x* old with the rank order of the variables in block 9 to determine the values of the variables. Record z* old and x* old in block 14. End the algorithm.

NOTE: If you had to convert the equation to a minimum equation then you must multiply the z* value by -1 to obtain the actual value of the equation.

The final step is simply identifying the solution derived by the algorithm. More importantly it defines what the solution set is and where to record the solution and solution set.

3.4 Algorithm 2 for Constraints.

This algorithm is applicable to all 0,1 variable constrained problems. It is a sub algorithm or step of the Overall Algorithm. Specifically, it is step 4 of the Overall Algorithm. This algorithm solves the system of constraints and derives a solution set and applies the solution set to the objective equation to derive a solution. Use the worksheet provided in Appendix D to record information. The number of each step corresponds to the number of the block on the worksheet.

3.4.1 Step 1

Are all constraints ≥? If no then multiply all constraints that are ≤ by -1. Multiply the constraints out. Example:

$$-2x_1(1-x_2)+x_3(1-x_4) \quad \geq 3 \quad \text{becomes} \; -2x_1+2x_1x_2+x_3-x_3x_4 \quad \geq 3$$

Record all constraints after this step in block 1.

This step is tied to step 1 of Algorithm 1 because step 1 of Algorithm 1 converts the objective equation to a minimization problem if it is not already that type of problem. Since the objective equation is a minimize problem then the constraints should be greater than or equal to inequalities as a matter of convention. The terms in the constraints are reduced to just cross products of variables and not cross products of $(1\text{-}x_1)$ because the algorithm uses the actual variables and not $(1\text{-}x_1)$.

3.4.2 Step 2

Multiply the i-th constraint by y_i (if y_i is the original variable then use z_i) and add the opposite of the right hand side (RHS) to both sides of the each constraint. Record in block 2. y_i is used to determine the tightness of the i-th constraint. It will be treated as a normal variable but will not be solved for.

The y variable is used as an indicator and does not return a real number to be used in solving the system of constraints. The author applied the logic that using the y_i ties all of the terms in all of the constraints into one constraint that should be greater than or equal to zero. It is similar to the concept of surrogate variables in geometric programming. If y_i for the i-th constraint is less than one then the i-th constraint has been found, by experience, to not restrict the objective equation or the constraint is not tight. The reason the author is adding the opposite of the y_i and not dividing by the RHS is based on the concept of all the algorithms. This concept is the term ratio and the variable ratio. Since the ratios that are used in establishing are reduced in magnitude by the number of variables in each term the ratios are further reduced in magnitude when you divide by the RHS. The term ratios are reduced so much that they become insignificant and it becomes difficult to determine which variable should be ranked first, second, etc. Chapter 5 discusses possible further research in this area.

3.4.3 Step 3

Record positive terms of all constraints in the block marked POS. Record the **absolute** value of the negative terms of all constraints in the block marked NEG. In the block marked RATIO record the ratio of the absolute value of the coefficient of each term over the number of variables, both original and y_i, in that term.

Step 3a

Record each variable in column 3a putting one variable in each row. Record all original variables first then the y_i.

Again, an administrative step to allow the user to organize the information.

3.4.4 Step 4

Scan each term in the positive block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio, below the term, in column 4 for the first variable.

NOTE: If the first variable is not present in any of the positive terms then record 0 for that variable in column 4.

NOTE: Do this for the y variables also.

Step 4a

Sum the recorded answers in column 4 for the first variable and record that value in column 4a of that variable row.

This step is the same as step 4 of Algorithm 1 except for the y_i variable. This step is also done for the y_i because these values will be used to determine the tightness of the i-th constraint.

3.4.5 Step 5

Repeat step 4 and 4a for each variable in column 3a. After this step you should have a real number recorded in column 4a for each variable in column 3a.

3.4.6 Step 6

Scan each term in the negative block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio below the term in column 6 of that variable row.

NOTE: If the first variable is not present in any of the negative terms then record 0 for that variable in column 6.

NOTE: Do this for the y variables also.

Step 6a

Sum the recorded answers in column 6 for the first variable and record that value in column 6a of that variable row.

3.4.7 Step 7

Repeat step 6 and 6a for each variable in column 3a. After this step you should have a real number recorded in column 6a for each variable in column 3a.

Steps 5 through 7 are the same as steps 5 through 7 of Algorithm 1. Again, the only difference is the y_i variable.

3.4.8 Step 8

For all variables other than the y_i introduced at Step 2. Create a ratio for each variable of the number recorded in column 6a over the number recorded in column 4a. Is there a zero in column 4a? If yes, then record infinity (i.e., ∞) for the ratio of that variable in column 8. Do this for all variables that have a zero in column 4a.

This step uses the same logic as step 8 of Algorithm 1.

3.4.8a Step 8a

For all variables y_i introduced at Step 2. If the ratio for y_i is less than one then that constraint in probably not restrictive. Record n (i.e., not restrictive) for the rank in block 9. If the ratio for y_i is greater than one then that constraint is restrictive. Record r (i.e., restrictive) for the rank in block 9. Restrictive is defined to mean that a particular combination of variables are turned on and the others are off in order to satisfy all the constraints. Non-restrictive means the original variables could be either 0 or 1 without violating the constraints.

Understanding that the ratio is the sum of the negative contributions divided by the sum of the positive contribution is the important concept of this step. If, for y_i, the ratio is less than one then this implies that there is greater positive contribution relative to the negative contribution. Contribution in this case is the contribution of each term toward satisfying the constraint. An example in Chapter 4 will demonstrate this concept in greater detail.

3.4.9 Step 9

In increasing order rank each variable according to the ratio created in column 8. The variable with the smallest ratio is ranked 1 block 9, etc. Record zero for the value of each variable in the value column of block 9.

NOTE: If more than one ratio equal to zero then rank those variables first among the other variables. Then rank them according to the decreasing values from column 4a. If more than one ratio equal to infinity then rank them last among the other variables. Then rank them according the increasing value from column 6a.

NOTE: If there are ties of non-zero or non-infinity ratios in column 8 then refer to column 4a and rank them in decreasing order.

NOTE: If there are ties in either 6a or 4a then break them arbitrarily.

The logic in rank ordering the variables in this step is the same as in step 9 of Algorithm 1.

3.4.10 Step 10

Record the value of 0 for the variables in block 10 as x* in vector form (i.e., (0,0,0,0)) in the rank order established in Step 9. Solve the system of constraints.

This step is significantly different from Algorithm 1 because Algorithm 1 sets the variables equal one and this algorithm sets them equal to zero. The logic here is understanding the variable ratio concept and the ranking concept. Remember that if a variable is ranked first then this implied that the positive contribution is greater than the negative contribution. By putting that variable first means that it will be changed to one first and consequently add positive terms to each of the constraints. The author understands that there will most likely be several non-linear terms and it may require more than one variable to add positive value to the constraints. Finally, there may be a variable that has positive contribution in all constraints except one but, this becomes insignificant when other variables are changed from zero to one.

3.4.11 Step 11

For each constraint does x* satisfy the constraints in block 1? If all constraints are satisfied then go to block 14. Else go to Step 12

This step is merely the iteration of the algorithm through the ranking of the variables until all the constraints are satisfied.

3.4.12 Step 12

Cross out any previous x* in block 10. Change the first zero variable in the rank order in block 9 to 1 by crossing out the 0 in the variable row of block 9 and record 1. Record x* new in block 10 in the order established in Step 9. Solve the system of constraints. For each constraint does x* new satisfy the constraints in block 1? If all constraints are satisfied then go to block 14. Else, go to step 13.

A continuation of the iteration through the ranking of the variables.

3.4.13 Step 13

Is there a zero variable remaining in block 9? If yes, then repeat step 12. If no, then end the algorithm because there is no feasible solution.

At this point all of the variables have been changed to one and the particular ranking of the variables did not produce a feasible solution. The order in which the variables are changed to one is the critical aspect of the algorithm. Referring to the binary tree in Chapter 2, the order in which each branch is chosen is paramount to obtaining a feasible, if not optimal solution.

3.4.14 Step 14

The current values of the variables in block 10, the x* column, is the solution set. Match x* with the rank order of the variables in block 9 to determine the values of the variables. Record x* in block 14. Use x* to solve for the value of the objective equation and record the value (z*) in block 14. End Algorithm 2. Return to Overall Algorithm.

This step is done to extract the final solution and solution set from the worksheet. Particular attention must be used because the final solution set is in increasing rank order and not in increasing subscript order. Although the solution set is written as a vector does not mean that the number listed in the third element of the vector is x_3.

Chapter 4
EXAMPLE PROBLEMS

4.1 Introduction.

In Chapter 3 the Overall Algorithm was discussed. Selected examples from Appendix A will be demonstrated in this chapter. The complete worksheets that the author will use in the solving the problems are in Appendix D.

4.2 Number 29.

The first problem will be solved using the worksheet and the code for Algorithm 1. The problem is an unconstrained objective equation from the book Integer Programming by Robert S. Garfinkel and George L. Nemhauser, 1972, page 362:

$$\max f(x) = 3x_1 - x_2 - 2x_1x_3x_5 + 2x_2x_6 - x_1x_4x_6 + 2x_4$$

4.2.1 Steps 1 thru 3 Algorithm 1.

The Overall Algorithm starts the process and moves to Algorithm 1 which is actually the only algorithm that applies to this unconstrainted problem. Referring to Chapter 3 on executing Steps 1 thru 3 of Algorithm 1 we have the first portion of the worksheet completed as shown in figure 4.1. Step 1 is convert the maximize problem to a minimize problem. Step 2 is simplify and in this case it was not needed so the equation from Step 1 is recorded in block 2. Finally, Step 3 which is group the positive terms together and the negative terms together.

1 Minimum Problem $-3x_1 + x_2 + 2x_1x_3x_5 - 2x_2x_6 + x_1x_4x_6 - 2x_4$	
2 $-3x_1 + x_2 + 2x_1x_3x_5 - 2x_2x_6 + x_1x_4x_6 - 2x_4$	
3 Group Terms	
POS.	NEG.
x_2 $2x_1x_3x_5$ $x_1x_4x_6$ $\frac{1}{1}$ $\frac{2}{3}$ $\frac{1}{3}$	$3x_1$ $2x_2x_6$ $2x_4$ $\frac{3}{1}$ $\frac{2}{2}$ $\frac{2}{1}$

4.1 Number 29, Algorithm 1, Steps 1-3 worksheet

4.2.2 Steps 3A thru 8 Algorithm 1.

Continuing the process we execute Steps 3a thru 8. Step 3a is recording the variables in column 3a of the worksheet. Step 4 is recording the positive term ratios in the variable row x_1 and column 4. The term ratio is recorded if the variable is present in that specific term. Step 4a is summing the values in column 4a and recording the sum in column 4a. Step 5 is repeat the process for the remaining variables in column 3a.

Likewise, Steps 6, 6a, and 7 are for the negative terms ratios. Finally, Step 8 is recording, for each variable, the number is column 6a over the number in column 4a and hence producing another ratio that will be used to rank the variables. We now have blocks 3a thru 8 completed as shown in figure 4.2.

3A Variables	(+) 4	4A	(-) 6	6A	(-) (+) 8
x_1	$\frac{2}{3}$ $\frac{1}{3}$	1	$\frac{3}{1}$	3	3
x_2	$\frac{1}{1}$	1	$\frac{2}{2}$	1	1
x_3	$\frac{2}{3}$	$\frac{2}{3}$	0	0	0
x_4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{1}$	2	6
x_5	$\frac{2}{3}$	$\frac{2}{3}$	0	0	0
x_6	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{2}$	1	3

4.2 Number 29, Algorithm 1, Steps 3a-8 worksheet

4.2.3 Steps 9 thru 14 Algorithm 1.

Using the values from column 8 we can now rank order the variables in increasing order. There are two ties among four entries in column 8. The first tie is between variables x_3 and x_5. According to step 9 these two variables would be ranked first among the other variables. Between variables x_3 and x_5 the variable that is ranked first is arbitrary because both variables are totally equal in all columns. Therefore, the first and second in the rank order is x_3 then x_5 in that order. The reader can verify that the order will not matter. The second tie is between variables x_1 and x_6. These are non-zero and non-infintiy ratios which means rank them in decreasing order according to column 4a. Hence, we have the rank order established as shown in figure 4.3.

9									
Rank	1	2	3	4	5	6	7	8	9
Variable	3	5	2	1	6	4			
Value	1	1	1	1	1	1			

4.3 Number 29, Algorithm 1, Step 9 worksheet

We have now established a rank order and value for all the variables. Setting all x_i equal to one we derive z*. O[illegible]rst z* will be z* old and its value is -3. We now change the value of the first variable, x_3, to 0 and derive a new z*. This z* will be called z* new and its value is -5. Moving to Step 12 we determine that z* new is less than z* old so, we replace z* old with z* new. Repeating Step 11 we cross out z* new, change the next variable, x_5, in the rank order to zero, and derive another z* new. The new z* new is -5 and z* old is not less than z* new therefore, we repeat Step 11 thru 12 one more time. The comparison in Step 12 this time tells us to move to Step 14. The final value of z* new is -4 and z* old is -5. Step 14 states that if we had to change our equation to minimization problem that we must multiply z* old by -1 to find the final solution. This gives us a final solution of 5 and a final solution set of $x_1 = x_2 = x_4 = x_6 = 1$ and $x_3 = x_5 = 0$ or (1,1,0,1,0,1). Figure 4.4 shows the applicable portions of the worksheet for steps 10 to 14. At Appendix F is the input required for number 29 and the output generated.

				10 z* old	x * old
				~~-3~~	(1,1,1,1,1,1)
11				~~-5~~	(0,1,1,1,1,1)
z* new	x* new	z* new	x* new	-5	(0,0,1,1,1,1)
~~-5~~	(0,1,1,1,1,1)				
-5	(0,0,1,1,1,1)				
-4	(0,0,0,1,1,1)				
				14 Solution z*= -5 x*= (1,1,0,1,0,1)	

4.4 Number 29, Algorithm 1, Steps 10-14 worksheet

4.3 Number 28.

The next problem is a constrained non-linear system of equations. The problem is from the book Integer Programming by Robert S. Garfinkel and George L. Nemhauser, 1972, page 363:

$$\max z(x) = -3x_1x_3x_5 - 2x_2 - 4x_2x_4 - 3x_5 \tag{4.1}$$

$$\text{s.t. } 2x_1 - 3x_1x_3 + x_4 - 2x_4x_5 \leq -2 \tag{4.2}$$

$$-x_1x_4 - x_2 + 2x_3 - x_4x_5 + 3x_2x_5 \leq 0 \tag{4.3}$$

The author will use the complete Overall Algorithm to solve the system of equations. Referring to either Chapter 3 or Appendix C, the flowcharts, we find that the first step of the Overall Algorithm is derive a solution and solution set with a rank order using Algorithm 1.

4.3.1 Steps 1, 2, and 3 of the Overall Algorithm and Algorithm 1.

Since Algorithm 1 was demonstrated in problem 29 it will be left to the reader to verify the solution to the objective equation. The solution for problem 28 is z* = 0 and a rank order is x_5, x_2, x_4, x_3, x_1. The value of all the variables is zero. It should be noted that Algorithm 1 allows more than one possible solution. This happens because there is a tie between variables x_1, x_3, and x_2, x_5 but, the change of the rank order does not change the solution. Once we have completed the steps of Algorithm 1 we return to the Overall Algorithm.

The Overall Algorithm leads us to apply the solution set derived in Algorithm 1 to the system of constraints. If they are satisfied, then end the algorithm. The solution set does not satisfy the system of constraints (eq. 4.1, eq. 4.2, eq. 4.3). As we continue the Overall Algorithm we change the first zero variable in the rank order to one and apply the new solution to the system of constraints. The new solution is $x_5 = x_2 = x_4 = x_3 = 0$ and $x_1 = 1$ which again does not satisfy the system of constraints. It can be shown that the solution set does not satisfy the system of constraints (eq. 4.1, eq. 4.2, eq. 4.3). Table 4.1 shows steps 2 and 3 of the Overall Algorithm. Step 2 checks each constraint using the current solution. Step 3 changes the last zero in the current solution to one then returns to step 2. The solutions listed in column step 3 are listed in rank order.

4.1 Number 28, Overall Algorithm, Steps 1-3

Step 3	Step 2	
Rank x_5, x_2, x_4, x_3, x_1 Initial values of $z^* = 0$	constraint 1 satisfied ?	constraint 2 satisfied ?
$(x_5, x_2, x_4, x_3, x_1)$		
(0,0,0,0,0)	0 no	0 yes
(0,0,0,0,1)	2 no	0 yes
(0,0,0,1,1)	-1 no	2 no
(0,0,1,1,1)	0 no	1 no
(0,1,1,1,1)	0 no	0 yes
(1,1,1,1,1)	-2 yes	2 no

Hence, we move to step 3 of the Overall Algorithm and use Algorithm 2 to solve the system of constraints. We will use the solution derived in Algorithm 1 of $z^* = 0$ and the rank order of x_5, x_2, x_4, x_3, x_1 to establish bounds of the solution in step 5 of the Overall Algorithm.

4.3.2 Steps 1, 2, and 3 of Algorithm 2.

Referring again to either Chapter 3 or Appendix C we find that step 1 is change all the $\leq$ to $\geq$, simplify, and record in block 1 of the worksheet. Step 2 is multiply the i-th constraint by y_i, add the opposite of the RHS to both sides of each constraint, and record the results in block 2. Step 3 is group all positive terms together and all negative terms together then record the absolute value of the term ratio. Figure 4.6 shows the completed worksheet.

1

$$-2x_1 + 3x_1x_3 - x_4 + 2x_4x_5 \geq 2$$

$$x_1x_4 + x_2 - 2x_3 + x_4x_5 - 3x_2x_5 \geq 0$$

2

$$-2x_1y_1 + 3x_1x_3y_1 - x_4y_1 + 2x_4x_5y_1 - 2y_1 \geq 0$$

$$x_1x_4y_2 + x_2y_2 - 2x_3y_2 + x_4x_5y_2 - 3x_2x_5y_2 \geq 0$$

3 Group Terms	
POS.	NEG.
$3x_1x_3y_1$ $\frac{3}{3}$ $2x_4x_5y_1$ $\frac{2}{3}$ $x_1x_4y_2$ $\frac{1}{3}$ x_2y_2 $\frac{1}{2}$ $x_4x_5y_2$ $\frac{1}{3}$	$2x_1y_1$ $\frac{2}{2}$ x_4y_1 $\frac{1}{2}$ $2x_3y_2$ $\frac{2}{2}$ $3x_2x_5y_2$ $\frac{3}{3}$ $2y_1$ $\frac{2}{1}$

4.5 Number 28, Algorithm 2, Steps 1-3 worksheet

4.3.3 Steps 3A Thru 9 of Algorithm 2.

Once we have established the term ratios and recorded the variables in column 3A, we are ready to record the ratios for each term in the appropriate column. If the variable is present in the positive term then record the ratio of that term in the variable row of column 4. This is step 4. Step 4A is summing these ratios in each variable row and recording the results in column 4A. Step 5 is completing this process for each variable, to include y_1 and y_2. In executing step 6, we apply the same concept to the negative terms and record the results in column 6. Again the process is completed for the negative terms. Step 6A is the summing of the ratios and step 7 is completing this process for each variable. The ratios are created in step 8 and are recorded in column 8. The rank order is done in step 9. Figure 4.7 shows the applicable portion of the completed worksheet.

3A	(+)		(-)				
	4	4A	6	6A	8	9	
x_1	$\frac{3}{3}\ \frac{1}{3}$	1.33	$\frac{2}{2}$	1	.75	2	0
x_2	$\frac{1}{2}$	.5	$\frac{3}{3}$	1	2	5	0
x_3	$\frac{3}{3}$	1	$\frac{2}{2}$	1	1	3	0
x_4	$\frac{2}{3}\ \frac{1}{3}\ \frac{1}{3}$	1.33	$\frac{1}{2}$	.5	.375	1	0
x_5	$\frac{2}{3}\ \frac{1}{3}$	1	$\frac{3}{3}$	1	1	4	0
y_1	$\frac{3}{3}\ \frac{2}{3}$	1.66	$\frac{2}{2}\ \frac{1}{2}\ \frac{2}{1}$	3.5	2.1	R	
y_2	$\frac{1}{3}\ \frac{1}{3}\ \frac{1}{2}$	1.16	$\frac{2}{2}\ \frac{3}{3}$	2	1.7	R	

4.6 Number 28, Algorithm 2, Steps 3a-9 worksheet

Before proceeding to the next steps it is interesting to note the values of y_i ratios. Both ratios are greater than 1 which indicates that the sum of the contributions of each term on the left hand side of each consrtaint is less than or equal to the RHS. Block 2 of figure 4.6 demonstrates this conjecture. Based on what the constant is on the RHS this could be a constraint that must be satisfied (tight) rather than a constraint that will always be satisfied.

4.3.4 Steps 10 Thru 14 of Algorithm 2.

Steps 10, 11, and 12 are the iterations through the constraints with a new x* each iteration. Specifically, step 10 derives the first solution with all variables equal to zero, and step 11 checks the solution set x*. Step 12 changes the first variable in the rank order to one and returns to check the solution set with the system of constraints to derive which constraint is violated. If no constraint is violated then we move to step 14. Otherwise, we continue changing variables to one until either a solution is found or no feasible solution is found. If no feasible solution is found then we are at step 13 and the algorithm is ended. If there is a solution found then we move to step 14. At step 14 we match x* with the rank order and derive a solution set, and ultimately the solution z*. Our final solution and solution set is shown is figure 4.8. We then return to the Overall Algorithm to finish the problem.

10 x*	(1,0,0,0,0)	(1,1,1,0,0)			
(0,0,0,0,0)	(1,1,0,0,0)	(1,1,1,1,0)			
14 Solution x* = (1,0,1,1,1)				z* = -6	

4.7 Number 28, Algorithm 2, Steps 10-14 worksheet

4.3.5 Steps 4 and 5 of the Overall Algorithm.

We now have two solutions and solution sets for system of equations. Steps 4 and 5 of the Overall Algorithm checks the solution against the system of constraints and establishes the upper and lower bounds of the optimal solution. These bounds are from Algorithm 1 and Algorithm 2 and they are 0 and -6. So, the optimal solution is conjectured to lie between these two points and in fact the optimal solution is -6.

Chapter 5
CONCLUSION AND SUGGESTIONS FOR FURTHER STUDY

5.1 Conclusion.

The Overall Algorithm contained in this thesis is not optimal, as demonstrated in Chapter 4. But the ease in which a solution is found demonstrates that it could be useful in providing a feasible starting solution and solution set for a more time consuming algorithm. More importantly it will identify a problem that has no solution. Algorithm 1 for the objective equation has been demonstrated to find the optimal solution regardless of the equation. Another important aspect of the Overall Algorithm is the ability to provide a least an upper and lower bound for the solution, if it exists, in a fraction of the time required to derive the optimum solution. Finally, since Algorithm 1 is coded a solution to the objective equation can be derived quickly and applied to the system of constraints. This will provide, at a minimum, a starting solution and solution set.

5.2 Further Study.

There are numerous areas of possible research and study. Several of these areas will be discussed in detail. One area not discussed in detail is researching the run time of this algorithm. If n were the number of variables then, other algorithms researched by the author required as much as 2^n possible iterations and the Overall Algorithm takes, at most, n iterations. Therefore, the Overall Algorithm is intuitively faster because it only makes one pass to find a solution rather than iterating to find all possible solutions. As the problems become larger how much faster is the run times compared to commercial software.

5.2.1 Proof of Optimality.

> In mathematics, "algorithm" is commonly understood to be an exact prescription, defining a computational process, leading from various initial data to the desired result (Markov, 1971).

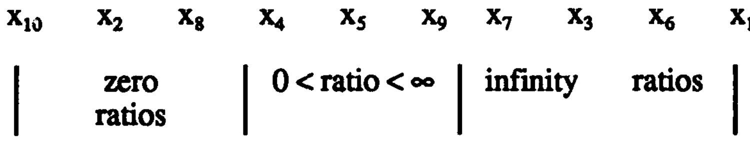
The author has demonstrated that the algorithm is exact. It has yet to be proven that the algorithm is exact. The following paragraphs outline how the author will prove the algorithm is exact or optimal.

First would be the proof of optimality with respect to Algorithm 1. The Overall Algorithm has been demonstrated to not be optimal (i.e., #11, #26, Appendix A) but, Algorithm 1 has always found the optimal solution when applied to either an unconstrained objective equation or a constrained objective equation where the minimum of the objective equation satisfied the constraints. A general outline of the proof could be to show that the solution derived is always contained in the optimal solution set. First would be the theorem for the partial solution set where the variables have ratios that are either equal to zero or infinity. The remaining portion of the proof would be to show that the remaining variables are part of the optimal solution set. Figure 5.1 shows an example of 10 variables in a rank order and each with a variable ratio. As one can see the rank order and ratios facilitates proving the complete solution set is contained in the optimal solution set. The first portion, as stated earlier, would contain the theorems that prove

1. The zero ratio variables, in the beginning of the rank order ,should be set to zero,

2. The infinity ratio variables, at the end of the rank order, should be set to one.

Next, would be the proof that the Overall Algorithm always provides bounds on the optimal solution, when it exists. As demonstrated the Overall Algorithm does provide bounds for the optimal solution and research conducted indicates that may always be true.


x_{10} x_2 x_8 x_4 x_5 x_9 x_7 x_3 x_6 x_1
zero ratios
$0 < \text{ratio} < \infty$
infinity ratios

5.1 Diagram of variable ratios

Finally, the Overall Algorithm is for a certain number of variables. If one could show optimality for a small number, less than 10, then it would be a simple step to show for large number of variables. Coding the Overall Algorithm would facilitate this area of research.

5.2.2 Code of the remaining Algorithms.

Another area of research is the coding of the Algorithm 2 and the Overall Algorithm. The Overall Algorithm and Algorithm 2 could be coded and linked together with Algorithm 1 which has already been coded. Once completed the researcher could conduct a comparison between the code produced and a published code. An example would be comparing solution sets and solutions between the code produced and STORM for the totally linear system of equations. Also, the current code needs to be expanded to accept problems with more than 30 variables and 30 terms. Based on the design of the code this would require changing the fixed memory allocation of 30 to a changin allocation to meet the size of the problem. Another modification of the code would be to improve the way data is input into the computer. Currently, for each term the user must answer yes the variable is present (enter 1) or no the variable is not present (enter 0) for each variable. This is more time consuming than running the code. If the user were only required to input the variables that were present in each term as the terms were input then this would reduce input time tremendously. Finally, an additional aspect of coding the complete algorithm is the research of the y_i variable.

5.2.3 Analysis of the y_i variable.

The specific area of further research is the analysis of the y_i variable used in Algorithm 2. The y_i variable seems unnecessary because the algorithm does not use the y_i variable directly. But application of the algorithm to example problems in Appendix A will demonstrate that the algorithm does not yield a optimal or feasible unless these y_i variables are used in the algorithm. The analysis could determine the relationship the y_i

and the RHS. The author considers this because the RHS usually has the most impact on the y_i. A review of the algorithm will show that the RHS is not multiplied by any other variable except the y_i and, therefore, usually has the most impact on the ratio of the y_i variable.

5.2.4 Division by the RHS.

Another area of research is dividing the right hand side (RHS) through both sides of the constraint instead of subtracting, from both sides of the constraint, the RHS. The reason the author is adding the opposite of the y_i and not dividing by the RHS is based on a concept in Algorithm 2. This concept is the term ratio and the variable ratio. Since the ratios that are used in establishing the term ratio are reduced in magnitude by the number of variables in each term the term ratios are further reduced in magnitude when you divide by the RHS. It is possible that the algorithm could be improved by only dividing each constraint by the RHS. The author has done several problems using this concept and has produced favorable results.

5.2.5 Improvement of the Overall Algorithm.

Another area related to the y variable is the fact that Algorithm 2 does not totally consider the constant RHS in the constraint. Currently the RHS is used in establishing the y_i ratio but it is not used in establishing any of the x_i ratios. Division by the RHS, as disussed, is a method of considering the RHS with the x_i variables. Another method would be to subtract the RHS from each coefficient. If the algorithm could be improved to consider RHS then this will move the solution set derived in Algorithm 2 closer to the optimum solution set and ultimately closer to the optimum solution.

5.3 Conclusion.

It is conjectured that the real class of problems that the Overall Algorithm can solve is not limited to the definition in Chapter 2. A method by Glover and Woolsey will replace any constraints that are non linear by constraints that are linear. The method is taught by Dr. Woolsey during his Integer Programming class and is explained in this fashion.

Consider the cross product x_1x_2 and let $x_1x_2 = x_{12}$. This cross product can now be replaced by the variable x_{12} and two constraints. The two constraints are:

$$x_1 + x_2 - x_{12} \leq 1$$
$$-x_1 - x_2 + 2x_{12} \leq 0$$

This will transform any problem that is not considered in the class of problems to a problem that the algorithm can solve. If minor further research could show this to be true then the Overall Algorithm will be applicable to any 0-1 variable problem.

REFERENCES CITED

Balas, Egon. 1965. An Additive Algorithm for Solving Linear Programs with Zero-one Variables. Operations Research. Volume 13, Number 4:517-546.

Emmons, Hamilton, et al. 1989. STORM Personal version 2.0. Oakland CA: HOLDEN-DAY.

Garfinkel, Robert S.; and Nemhauser, George L. 1972. Integer Programming. New York: John Wiley & Sons.

Glover, F.; and Woolsey R. E. D. 1974. JORSA. Volume 22, Number 1:181.

Glover, Fred; and Zionts Stanley. 1965. A Note on the Additive Algorithm of Balas. Operations Research. Volume 13, Number 4:546-549.

Hammer, Peter L.; and Rudeanu, Sergiu. 1967. Pseudo-Boolean Programming. Operations Research. Volume 17, Number 2:233-261.

Hammer, Peter L.; and Rudeanu, Sergiu. 1968. Boolean Methods in Operations Research. New York: Springer-Verlag New York.

Hammer, Peter L. 1974. Boolean Elements in Combinatorial Optimization: a survey. Part I in Combinatorial Programming: Methods and Applications. Ed. B. Roy. Boston: D. Reidel Publishing Company.

Lee, Sang M.; Moore, Laurence J.; And Taylor, Bernard W. 1985. Management Science. 2nd ed. Boston: Allyn and Bacon.

Markov, A. A. 1971. Theory of Algorithms. Jerusalem: Keter Press.

Press, William H., et al. 1988. Numerical Recipes in C. New York: Cambridge Univ. Press.

Taha, Hamdy A. 1975. Integer Programming. New York: Academic Press.

Winston, Wayne L. 1987. Operations Research: Applications and Algorithms. Boston: PWS-KENT Publishing Company.

Zionts, Stanley. 1974. Linear and Integer Programming. New Jersey: Prentice-Hall.

APPENDIX A
PROBLEMS

Below are several problems with the answers provided. The format for the problems is: problem, problem type (N-nonlinear constraints or objective equation, U-unconstrained, L-linear objective equation and constraints), how the answer was obtained (i.e., storm, IE-implicit enumeration, Balas method, BR-book referenced, etc), z* value and, variables values in vector form that obtains the z* value. All variables are 0,1. All problems not referenced were generated by the author. At the end of the appendix are the solutions derived by the algorithm for all problems.

Author Kevin J. Loy
updated February 26, 1991

1. MIN $x_2 - x_1 - x_2x_1$
   U IE -1 (1,1),(0,1)

2. MAX $-2x_1 + 3x_2 - 5x_1x_2$
   U IE 3 (0,1)

3. MAX $xyz - 7x + 6y - 4z$
   U IE 6 (0,1,0)

4. MAX $100x_1 - 200x_1x_2 + 150x_2$
   U IE 150 (0,1)

5. Min $-25x_1 - 30x_1x_2 - 3x_2$
   U IE -58 (1,1)

6. MIN $x_2 - x_1 - x_2x_1$
   s.t. $x_2 - x_1 \leq 0$
   N IE -1 (1,1)

7. MAX $-2x_1 + 3x_2 - 5x_1x_2$
   s.t. $2x_1 - x_2 \leq 1$
   N IE 3 (0,1)

8. Min $4x_1x_2 - 3x_2 + 2x_3x_2 - 5x_1x_3$
s.t. $x_1 + x_2 \geq 1$

$x_2 + x_3 \geq 1$

N IE -5 (1,0,1)

9.
Minimize $5x_1 + 3x_2 + x_3 - 9x_1x_2x_3$

s.t. $x_1 + x_2 + x_3 \leq 1$

$x_2 + x_3 \geq 1$

N IE 3 (0,0,1)

10. (Lee, Moore, Taylor, 1985, 755).
Minimize $Z = 2x_1 + x_2 + 3x_3 + 2x_4 + 4x_5$

s.t.

$x_1 + 2x_2 + x_3 + x_4 + 2x_5 \geq 4$

$7x_1 + x_2 - 3x_4 + 3x_5 \geq 2$

$-3x_1 + 3x_2 + 2x_3 - x_5 \geq -1$

L BR 5 (0,1,0,0,1),(1,1,0,1,0)

11. (Lee, Moore, Taylor, 1985, 772).
Miminize $Z = 30x_1 + 12x_2 + 10x_3 + 18x_4$

s.t.

$2x_1 + 4x_2 + 6x_3 - 2x_4 \geq 4$

$4x_1 + x_2 - x_3 - 2x_4 \geq 3$

L STORM 40(1,0,1,0)

12. (Lee, Moore, Taylor, 1985, 772).
Minimize $Z = 6x_1 + 3x_2 + 9x_3 + 6x_4 + 12x_5$

s.t.

$3x_1 + 6x_2 + 3x_3 + 3x_4 + 6x_5 \geq 12$

$21x_1 + 3x_2 \quad - 9x_4 + 9x_5 \geq 6$

$-9x_1 + 9x_2 + 6x_3 \quad - 3x_5 \geq -3$

L STORM 15(1,1,0,1,0)

13. (Winston, 1987, 364).
Max $z = x_1 - x_2$

s.t.

$x_1 + 2x_2 \leq 2$

$x_1 - x_2 \leq 1$

L IE 0 (0,0)

14.
Minimize $2x_1x_3 - 19x_4x_2x_1 + 15x_3x_2 - 9x_3x_4$

s.t.

$2x_2x_4 - 5x_1x_4 + 6x_3x_2 \leq 4$

$x_2x_3 - 4x_1 + x_1x_4 + 3x_3 \leq 5$

$2x_3 + 5x_3x_2x_1 - 4x_4 \geq 3$

N IE -11 (1,1,1,1)

15.

Maximize $2x_3 - 4x_1x_2 - 3x_2x_3 + 8x_3x_4 - 6x_4$

s.t.

$x_1x_2 - x_3x_1 + 2x_2x_4 \geq 1$

$2x_3x_4 - 5x_2x_4 + 4x_1x_3 \geq 1$

N IE -3 (1,1,1,1)

16.

Minimize $-4x_2 + 2x_3 - 3x_1x_3 - 5x_1x_4$

s.t.

$3x_1x_3 - 1x_3x_4 + 2x_1x_3x_4 \geq 3$

$-2x_3x_4 - 3x_2x_4 + 2x_2x_3 + x_1 \geq -2$

N IE -10 (1,1,1,1)

17.

Maximize $-3x_2x_4 + 4x_3x_4 - x_1x_3 + 4x_2x_3 + x_2x_3x_4$

s.t.

$-x_2x_3 + 3x_4 + x_1x_3 - 2x_3x_4 \leq 0$

$5x_1x_4 - x_2x_3 + 3x_1x_3x_4 \leq 5$

$x_1x_2 + x_2x_3 - x_2x_4 \leq 1$

N IE 6 (0,1,1,1)

18.

Maximize $5x_1x_3 - 2x_1x_3x_4 + 4x_2x_3x_4 - 3x_3x_4 + 4x_2x_4$

s.t.

$$6x_3x_4 + x_1x_3x_4 + 3x_1x_2 \le 6$$

$$-2x_2x_3 + 5x_1x_4 - 2x_1x_2x_4 + 5x_3x_4 \le 5$$

$$x_1x_2 + 2x_2x_3 - x_2 - x_1x_2x_3 + 2x_3x_4 \le 2$$

N IE 5 (1,0,1,0),(1,1,1,0)

19.

Minimize $5x_1x_3 - 3x_2x_3 + 2x_1 - x_2x_1$

s.t.

$$3x_1x_3 - 2x_3 + 2x_2x_3 + 2x_2 \ge 2$$

$$2x_1x_2 + 4x_2x_3 - 3x_3 + x_2 - 3x_1x_3 \le 2$$

N IE -3 (0,1,1)

20.

Minimize $10x_2x_3x_4 - 8x_2x_4 + 3x_1x_3 - 7x_3x_4$

s.t.

$$x_1x_2 - x_2x_3 + x_3x_4 - x_1x_3 + 2x_1x_4 \ge 2$$

$$2x_1x_3 + x_3x_4 + x_2x_4 - 3x_2x_3 - 2x_3 - 2x_1x_4 \le 3$$

N IE -8 (1,1,0,1)

21.
Maximize $11x_2x_3 - 4x_1x_2 + 3x_1x_3 - 2x_2 + x_1$

s.t.

$$4x_1x_2 - 2x_3 + 4x_1x_3 - x_2x_3 \le 4$$

$$2x_1x_3 - 3x_1 + 3x_2 + 2x_3 - x_2x_3 \le 2$$

$$x_2x_3 + x_1x_3 - x_1x_2 - x_1x_2x_3 \ge 1$$

N IE 4 (1,0,1)

22. (Winston, 1987, 419).
Maximize $4x_1 + 2x_2 - x_3 + 2x_4$

s.t.

$$x_1 + 3x_2 - x_3 - 2x_4 \ge 1$$

L IE 8 (1,1,0,1)

23. (Winston, 1987, 424).
Max $z = 2x_1 - x_2 + x_3$

s.t.

$$x_1 + 2x_2 - x_3 \le 1$$

$$x_1 + \quad x_2 + x_3 \le 2$$

L BR 3 (1,0,1)

24. (Winston, 1987, 429).

Max $z = 5x_1 - 7x_2 + 10x_3 + 3x_4 - x_5$

s.t.

$$-x_1 - 3x_2 + 3x_3 - x_4 - 2x_5 \geq 0$$

$$-2x_1 - 5x_2 + 3x_3 - 2x_4 - 2x_5 \leq 3$$

$$-x_2 + x_3 + x_4 - x_5 \geq 2$$

L BR 18 (1,0,1,1,0)

25. (Balas, Operations Research, 1965, 13:536).
$-5x_1 + 7x_2 + 10x_3 - 3x_4 + x_5 =$ Min

$$-x_1 - 3x_2 + 5x_3 - x_4 - 4x_5 \geq 0$$

$$-2x_1 - 6x_2 + 3x_3 - 2x_4 - 2x_5 \leq -4$$

$$-2x_2 + 2x_3 + x_4 - x_5 \geq 0$$

L Balas' Method9(1,1,1,1,0)

26. (Hammer and Rudeanu, Operations Research, 1967, 13:254).
Min $7x_1 - 2x_2 + 3x_3 + 2x_4 - x_5 - 6x_6 - 4x_7 + 2x_2(1 - x_5) - 5(1 - x_2)x_7$

$+4x_3x_7 + (1 - x_5)(1 - x_7) + 3x_6x_7 + x_1(1 - x_3)x_5 + 4x_1(1 - x_3)(1 - x_6)$

$-5x_1x_2x_4 + x_2x_5x_7 - 3x_3x_5x_7 + 2x_3(1 - x_5)x_6(1 - x_7)$

s.t.

$$2(1 - x_1) - 5x_2 + 3x_3 + 4(1 - x_4) - 7x_5 + 16x_6 - x_7 \geq -4$$

$$x_1x_2 + 4(1 - x_1)x_3 - 3x_2x_3x_5 + 6(1 - x_2)x_4x_6 \geq -1$$

$$3x_2x_4 - 5(1 - x_1)(1 - x_3)(1 - x_5) + 4x_4x_6 \geq 1$$

N H&R's Method-11(0,0,0,1,1,1,1)

27. (Garfinkel and Nemhauser, 1972, 349, eq[19]).
    Unconstrained Maximization

    $f(x) = 2x_1 + x_2 - 7x_3 - 5x_1x_2x_3 + 3x_2x_4 + 9x_4x_5$

    U Balas' Method15(1,1,0,1,1)

28. (Garfinkel and Nemhauser, 1972, 362).
    max $z(x) = -3x_1x_3x_5 - 2x_2 - 4x_2x_4 - 3x_5$

    s.t.

    $$2x_1 - 3x_1x_3 + x_4 - 2x_4x_5 \le -2$$

    $$-x_1x_4 - x_2 + 2x_3 - x_4x_5 + 3x_2x_5 \le 0$$

    N IE -6 (1,0,1,1,1,)

29. (Garfinkel and Nemhauser, 1972, 363).
    max $f(x) = 3x_1 - x_2 - 2x_1x_3x_5 + 2x_2x_6 - x_1x_4x_6 + 2x_4$
    U IE 5 (1,0,0,1,0,0),(1,1,0,1,0,1)

30. (Hammer and Rudeanu, 1968, 104).
    minimize $2 + 3x_1 - 2x_2 - 5x_3 + 2x_4 + 4x_6$

    s.t.

    $$2x_1 - 3x_2 + 5x_3 - 4x_4 + 2x_5 - x_6 \le 2$$

    $$4x_1 + 2x_2 + x_3 + 8x_4 - x_5 - 3x_6 \ge 4$$

    L H&R's Method-3(0,1,1,1,0,0),(0,1,1,1,1,0)

31. (Hammer and Rudeanu, 1968, 110).

Min f $= 2 - 3x_1 + x_2 + 5x_3$

s.t.

$2x_1 + 3x_2 + 5x_3 \geq 3$

$4x_1 + 5x_2 - 3x_3 \geq 1$

L H&R's Method0(1,1,0)

32. (Hammer and Rudeanu, 1968, 117).
Min f $= 2x_1 + 3x_2 - 7x_3 - 5x_1x_2x_3 + 3x_2x_4 + 9x_4x_5 - 2x_1x_5$

N H&R's Method-9(1,1,1,0,1)

33. (Hammer and Rudeanu, 1968, 118).
Min f $= 2x_1 + 3x_2 - 7x_3 - 5x_1x_2x_3 + 3x_2x_4 + 9x_4x_5$
N H&R's Method-7(0,0,1,0,0),(0,0,1,0,1),(0,0,1,1,0),(1,1,1,0,0),(1,1,1,0,1)

34. (Hammer and Rudeanu, 1968, 126).
minimize $3x_1(1-x_2) - 8(1-x_1)x_3x_6 + 4x_2x_5(1-x_6) - 7(1-x_5)x_6 + 3x_4 - 5x_4x_5x_6$

s.t.

$2x_1 - 3x_2 + 5x_3 - 4x_4 + 2x_5 - x_6 \leq 2$

$4x_1 + 2x_2 + x_3 + 8x_4 - x_5 - 3x_6 \geq 4$

N H&R's Method-12(0,1,1,1,0,1),(0,0,1,1,0,1)

35. (Hammer and Rudeanu, 1968, 136).

Min f $= 3x_1x_2 + 9x_2x_3x_4 - 7(1-x_1)x_5x_6 + 2x_3x_4(1-x_6) + 4x_1(1-x_2)x_3(1-x_4)(1-x_5)x_6 - 5x_7 + 5x_8 + 2x_7(1-x_8)$
U H&R's Method-10(0,1,0,-,1,1,1,0),(0,1,1,0,1,1,1,0)

36. (Hammer and Rudeanu, 1968, 138).

min $2x_1x_2 - 3x_1x_4 - 5x_2 - 8x_2x_3x_4 - 3x_2x_3 + 2x_3x_6 - 5x_4x_6 + 7x_5x_6x_7 - 4x_7x_8 + 2x_4x_6x_8$
U H&R's Method-10(1,1,1,1,0,1,0,0)

37. (Hammer and Rudeanu, 1968, 148). Locally minimizing points find

$f(x_1,x_2,\ldots)=3x_1(1-x_2)-8(1-x_1)x_3x_6+4x_2x_5(1-x_6)-7(1-x_5)x_6+3x_4-5x_4x_5x_6$

s.t.

$2x_1-3x_2+5x_3-4x_4+2x_5-x_6\le 2$

$4x_1+2x_2+x_3+8x_4-x_5-3x_6\ge 4$

N H&R's Method-12(0,-,1,1,-,1)

38. (Taha, 1975, 118).
Min z(y) $=4y_1y_3y_4+6y_3y_4y_5+12y_1y_5-2y_1y_2-8y_1y_3$

s.t.

$8y_1y_2+4y_1y_3y_4+y_3y_4y_5+y_1y_3-5y_1y_5\le 4$

$6y_1y_2+3y_1y_3y_4+2y_3y_4y_5-y_1y_5\le 4$

$-2y_1y_2-9y_1y_3y_4-3y_3y_4y_5-2y_1y_3-3y_1y_5\le -8$

N Lawler & Bell-4(1,0,1,1,0)

39. (Taha, 1975, 133).
minimize $z=-5y_1+7y_2+10y_3-3y_4+y_5$

s.t.

$-y_1-3y_2+5y_3-y_4-4y_5\ge 0$

$-2y_1-6y_2+3y_3-2y_4-2y_5\le -4$

$-y_2+2y_3+y_4-y_5\ge 2$

L STORM 9(1,1,1,1,0)

40. (Taha, 1975, 136).

minimize $z = 75x_1 + 100x_2 + 3x_4 + 4x_5 - 100x_1x_2 - 200x_1x_3 - 400x_2x_3 - 4x_4x_5 - 8x_4x_6 - 16x_5x_6$

U IE -546(1,1,1,1,1,1)

41. (Taha, 1975, 136).

minimize $z = 75x_1 + 100x_2 + 3x_4 + 4x_5 - 100x_1x_2 - 200x_1x_3 - 400x_2x_3 - 4x_4x_5 - 8x_4x_6 - 16x_5x_6$

s.t.

$$x_1 + 2x_2 + 4x_3 + x_4 + 2x_5 + 4x_6 \geq 2$$

$$x_1 + 2x_2 + 4x_3 - x_4 - 2x_5 - 4x_6 \geq -2$$

$$-x_1 - 2x_2 - 4x_3 - x_4 - 6x_5 + 12x_6 \geq -6$$

N IE -546(1,1,1,1,1,1)

42. (Taha, 1975, 137).

minimize $z = x_1x_7 + 3x_2x_6 + x_3x_5 + 7x_4$

s.t.

$$x_4 + x_5 + 6x_6 \geq 8$$

$$3x_1x_3 + 6x_4 + +4x_5 \leq 20$$

$$4x_1 + 2x_3 + x_6x_7 \leq 15$$

N IE 7 (1,0,0,1,1,1,0)

NOTE: Reference problem modified from to allow feasible solution.

43. (Zionts, 1974, 443).

minimize $z = 5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5$

s.t.

$-x_1 + 3x_2 - 5x_3 - x_4 + 4x_5 \le -2$

$2x_1 - 6x_2 + 3x_3 + 2x_4 - 2x_5 \le 0$

$x_2 - 2x_3 + x_4 + x_5 \le -1$

L Balas' Method17(0,1,1,0,0)

44. (Zionts, 1974, 465).
Minimize $z = 3x_1 + 7x_2 + 9x_3 + 2x_4 + 6x_5$

s.t.

$3x_1 + x_2 + 4x_3 + 5x_4 + 8x_5 \ge 10$

$5x_1 + 2x_2 + 9x_3 + x_4 + 5x_5 \ge 12$

$3x_1 + x_2 + 3x_3 + 2x_4 + 2x_5 \ge 2$

L STORM 14(1,0,1,1,0)

45. (Hammer, 1975, 74).
minimize $f = -x_1 + 3x_2 + x_1x_4 - 3x_1x_3 + 2x_2x_4 + 3x_3x_4 - 4x_2x_3$
U Hammer's Algorithm-5(1,1,1,0)

46. (Garfinkel and Nemhauser, 1972, 347).
max $z(x) = -2x_1x_3 - 4x_2 - 3x_1x_5 - 2x_4 - 3x_5$

s.t.

$-x_1x_4 - 3x_2x_5 - (-x_1 - 2x_3 - x_4x_5 - 1) \le 0$

$-2x_1 - 5x_3 - (-2x_4 - 3x_5 - 2) \le 0$

N Lexicographic Enum. Algorithm-7 (0,1,1,0,1)

47. (Taha, 1975, 105).

minimize $z = 3x_1 + 2x_2 + 5x_3 + 2x_4 + 3x_5$

s.t.

$$-x_1 - x_2 + x_3 + 2x_4 - x_5 \leq 1$$

$$-7x_1 \quad + 3x_3 - 4x_4 - 3x_5 \leq -2$$

$$11x_1 - 6x_2 \quad - 3x_4 - 3x_5 \leq -1$$

L Balas' Additive Algorithm3(0,0,0,0,1)

48. (Hammer and Rudeanu, 1968, 104).

minimize $2 + 3x_1 - 2x_2 - 5x_3 + 2x_4 + 4x_5$

s.t.

$$x_1x_2 + 4(1 - x_1)x_3 - 3x_2x_3x_5 + 6(1 - x_2)x_4x_6 \geq -1$$

$$3x_2x_4 - 5(1 - x_1)(1 - x_3)(1 - x_5) + 4x_4x_6 \geq 1$$

N H&R's Method-3(0,1,1,1,0,0),(0,1,1,1,1,0)

49. Problem 11, I.P. Exam 1989.

Find the minimum of $f = 2x_1x_2 - 3x_2x_4 + 9x_4x_6 + 5x_3x_5x_6 - 7x_3x_5 - 5x_1x_4x_6 - 3x_1x_3 - 13x_6$

U Balas' Algorithm-18(1,0,1,0,1,1)

In the tables that follow are the optimum solutions and the Overall Algorithm solutions. Algo 1 is the solution derived from Algorithm 1 and algo 2 is the solution derived from Algorithm 2.

A-1 Comparison of solutions.

Problem number	algorithm value algo1	algorithm value algo2	optimum value	percent near optimum
1/U	-1		-1	100
2/U	3		3	100
3/U	6		6	100
4/U	150		150	100
5/U	-58		-58	100
6/U	-1		-1	100
7/U	3		3	100
8/U	-5		-5	100
9/U	3		3	100
10/L	5		5	100
11/L	42	0	40	
12/L	15		15	100
13/L	1		1	100
14/N	-11		-11	100
15/N	-3		-3	100
16/N	-10		-10	100

(continued)

A-1 Comparison of solutions (continued).

Problem number	algorithm value		optimum value	percent near optimum
	algo 1	algo 2		
17/N	6		6	100
18/N	5		5	100
19/N	-3		-3	100
20/N	-8		-8	100
21/N	4		4	100
22/L	-8		-8	100
23/L	-3		-3	100
24/L	-18		-18	100
25/L	-8		-8	100
26/N	-7	-13	-11	
27/U	15		15	100
28/N	-6		-6	100
29/U	5		5	100
30/L	-3		-3	100
31/L	0		0	100
32/U	-9		-9	100
33/U	-7		-7	100
34/N	-12		-12	100

(continued)

A-1 Comparison of solutions (continued).

Problem number	algorithm value algo 1 algo 2	optimum value	percent near optimum
35/U	-10	-10	100
36/U	-22	-20	100
37/N	-12	-12	100
38/N	14 -10	-4	
39/L	9	9	100
40/U	-546	-546	100
41/N	-546	-546	100
42/N	12*	7	
43/L	17	17	100
44/L	18*	14	
45/U	-5	-5	100
46/N		-7	
47/L	4*	3	
48/N	-3	-3	100
49/U	-18	-18	100

* = solution was derived in step 3 of Overall Algorithm

APPENDIX B
CODE OF ALGORITHM 1

```
/***********************************************************************/
/*                Colorado School of Mines                 */
/*          Mathematics and Computer Science Department          */
/*                   Author: Kevin J. Loy             */
/*          written as part of thesis on 8 November 1990.        */
/* This program calculates the minimum value of a Pseudo-Boolean equation.*/
/* It assumes: 1- the equation is multiplied out (i.e., 2x(1-y)=2x-2xy)  */
/*         2- every term has at least one variable            */
/*         3- there are no more than 30 terms and 30 variables     */
/*         4- the equation is a minimum equation             */
/*                                  */
/*  It can be modified to accept more variables or terms by:          */
/*          1-changing ROW to 1+ the number of terms          */
/*          2- changing COL to 1+ the number of variables.       */
/*                                  */
/***********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define ROW 31
#define COL 31

void indexx();
void *alloc1(size_t n1, size_t size);
void **alloc2(size_t n1, size_t n2, size_t size);
void free1(void *p);
void free2(void **p);

main()

{
/* counters for the various for loops */
int i,j,k,l,m,a,b,c;

/* number of terms and number of variables  */
int nterms, nvars, a_ct, b_ct, c_ct, answer, max;

/*  Arrays for the terms, their ratios, the counters that count the       */
/*  number of variables in each term, sum of positive and negative con-   */
/*  tribution, the variable ratios, and the 3 groups of rankings        */

float **terms;
float **output_eq;
float *term_ratios;
float *count;
float *pos;
float *neg;
float *var_ratios;
float *group_a;
float *group_b;
float *group_c;
float z_old;
float z_new;
int *rank;
int *rank_a;
int *rank_b;
int *rank_c;
float *sol;

/*        construct memory through dynamic memory allocation        */
term_ratios = (float*)alloc1( ROW, sizeof(float));
count = (float*)alloc1( ROW, sizeof(float));
pos = (float*)alloc1( COL, sizeof(float));
neg = (float*)alloc1( COL, sizeof(float));
var_ratios = (float*)alloc1( COL, sizeof(float));
group_a = (float*)alloc1( COL, sizeof(float));
group_b = (float*)alloc1( COL, sizeof(float));
group_c = (float*)alloc1( COL, sizeof(float));
rank = (int*)alloc1( COL, sizeof(int));
rank_a = (int*)alloc1( COL, sizeof(int));
rank_b = (int*)alloc1( COL, sizeof(int));
rank_c = (int*)alloc1( COL, sizeof(int));
sol = (float*)alloc1( COL, sizeof(float));
terms = (float**)alloc2( COL, ROW, sizeof(float));
output_eq = (float**)alloc2( COL, ROW, sizeof(float));

printf(" \n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
printf("
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
printf("~~~~~~~~~~~~#\n");
printf("     # This program solves Pseudo-Boolean unconstrained Equat");
printf("ions.    #\n");
printf("     # It assumes:                              ");
printf("      #\n");
printf("     #   1. The equation is multiplied-out(i.e.,2x1(1-x2)=2");
printf("x1-2x1x2) #\n");
printf("     #   2. The equation is a minimization or maximization ");
printf("      #\n");
printf("     #   3. Every term has at least one variable          ");
printf("      #\n");
printf("     #   4. No more than 30 terms and 30 variables          ");
printf("      #\n");
printf("     #   5. The person inputting data can read.          ");
printf("      #\n");
```

```
printf("      # The input that can be changed is coefficients and whet");
printf("her     #\n");
printf("      # or not variable Xi is present if 0 or 1 is not entered");
printf(".      #\n");
printf("      #                                          ");
printf("       #\n");
printf("      #          The author is:   Kevin J. Loy          ");
printf("       #\n");
printf("
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
printf("~~~~~~~~~~~~\n\n\n\n\n\n\n\n\n");


printf("If equation is a maximum equation then enter -1. Else, enter 1.\n");
printf("Do not enter anything else but these two numbers.\n");
scanf("%d",&max);
while((max != 1)&&(max != -1))
      {
      printf("If you do not enter the correct number the program will");
      printf(" not run!\n");
      printf("If equation is a maximum equation then enter -1.\n");
      printf("Else, enter 1\n");
      scanf("%d",&max);
      }
printf("Input each term one at a time. Input the coefficient first with\n");
printf("the appropriate sign. If the variable is present in that term\n");
printf("then enter 1 to signify that variable is in that term. Else\n");
printf("enter 0 to signify that the variable is not present.\n\n\n");

/* Initialize all variable and arrays to 0. */
nterms = nvars = z_old = z_new = answer = 0;

/*                  prompt the user for input                    */
printf("How many terms are there?\n");
scanf("%d",&nterms);

while(( nterms <= 1) || ( nterms > 31))
      {
      printf("Please re-input numbers of terms.  It must be greater ");
      printf("than 1 and less than 31.\n");
      scanf("%d",&nterms);
      }
printf("How many variables are there?\n");
scanf("%d",&nvars);
```

```
while(( nvars <= 1) || ( nvars > 31))
      {
      printf("Please re-input numbers of variables that is ");
      printf("greater than 0 and less than 31.\n");
      scanf("%d",&nvars);
      }
for (i=0; i < nterms + 1; i++)
      {
      term_ratios[i] = 0;
      count[i] = 0;
      for( j=0; j < nvars + 1; j++)
         {
         terms[i][j] = 0;
         output_eq[i][j] = 0;
         }
      }
for (j=0; j < nvars + 1; j++)
      {
      pos[j] = 0;
      neg[j] = 0;
      var_ratios[j] = 0;
      rank[j] = 0;
      sol[j] = 1;
      rank_a[j] = 0;
      rank_b[j] = 0;
      rank_c[j] = 0;
      group_a[j] = 0;
      group_b[j] = 0;
      group_c[j] = 0;
      }
/*              Input values of equations.                  */
/*     If variable is present then 1 is entered.  Else 0 is entered.   */
for (i=1; i <= nterms; i++)
      {
      printf("What is the coefficient of term %d?\n",i);
      scanf("%f",&terms[i][0]);
      for(j=1; j <= nvars; j++)
         {
         printf("If variable %d is in term %d then enter 1\n",j,i);
         printf("Else enter 0\n");
         scanf("%f",&terms[i][j]);
         }
      }

/*             Verify input of coefficients                 */
printf("The coefficient of each term in order are %.0f",terms[1][0]);
for( i=2; i<= nterms; i++)
      printf(", %.0f",terms[i][0]);
printf("\nIf this correct, enter 1\n");
printf("Else, enter 0 and re-enter all coefficients.\n");
scanf("%d", &answer);
if (answer == 0)
      for( i=1;i <= nterms; i++)
          {
          printf("What is the coefficient of term %d?\n",i);
          scanf("%f", &terms[i][0]);
          }
/*    Check input of variables to ensure they are all either 0 or 1.     */
for(i=1; i<= nterms; i++)
      {
      for( j=1; j <= nvars; j++)
          {
          if ((terms[i][j] != 0 ) && (terms [i][j] != 1 ))
              {
              printf("Variable %d in term %d is not correct.\n",j,i);
              printf("Please re-input.\n");
              scanf("%f",&terms[i][j]);
              }
          }
      }
/*                     Copy for output                        */
for(i=1; i <=nterms; i++)
      {
      for(j=0; j <= nvars ; j++)
          {
          output_eq[i][j] = terms[i][j];
          }
      }

/*    Convert to minimize problem to solve.  If, needed.            */
for(i=1; i <= nterms; i++)
      terms[i][0] = terms[i][0] * max;

/* Sum number of variables in each term. */
for (i=1; i <= nterms; i++)
      {
      for (j=1; j <= nvars; j++)
          count[i] = count[i] + terms[i][j];
      }

/* Step 2.  Create absolute value ratio for each term. Else stmt for term */
/* is input that contains no variables (i,e,. all 0's input ).          */
for (i=1; i <= nterms; i++)
      {
      if (count[i] != 0)
          {
```

```
          term_ratios[i] = terms[i][0]/count[i];
          term_ratios[i] = fabs( term_ratios[i] );
          }

       else
          {
          printf("Term %d has no variables.  Please re-input.\n\n",i);
          for(j=1; j <= nvars; j++)
            {
            printf("If variable %d is in term %d then enter 1\n",j,i);
            printf("Else enter 0\n");
            scanf("%f",&terms[i][j]);
            output_eq[i][j] = terms[i][j];
            }
          for( j=1; j <= nvars; j++)
            {
            if ((terms[i][j] != 0 ) && (terms [i][j] != 1 ))
              {
              printf("Variable %d in term %d is not correct.\n",j,i);
              printf("Please re-input at least 1 variable");
              scanf("%f",&terms[i][j]);
             output_eq[i][j] = terms[i][j];
              }
            }
          for (j=1; j <= nvars; j++)
            count[i] = count[i] + terms[i][j];
          term_ratios[i] = terms[i][0]/count[i];
          term_ratios[i] = fabs( term_ratios[i] );
          }
       }

/* Steps 4 thru 7. Scanning positive terms and negative terms and adding  */
/* ratio of term if variable is present.                                  */
for (i=1; i <= nterms ; i++)
       {
       if (terms[i][0] < 0)
          {
          for( j=1; j <= nvars; j++)
            {
            if (terms[i][j] == 1)
              neg[j] = neg[j] + term_ratios[i];
            }
          }
       if (terms[i][0] > 0)
          {
          for( j=1; j <= nvars; j++)
              {
              if (terms[i][j] == 1)
```

```
                pos[j] = pos[j] + term_ratios[i];
             }
          }
       }
/* Step 8. Creating ratio for variables.                    */
for (j=1; j <= nvars; j++)
       {
       if (pos[j] != 0)
          var_ratios[j] = neg[j]/pos[j];

       else
          var_ratios[j] = 10000000.0;
       }
/*  put ratios into 3 group for sorting                     */
a_ct = b_ct = c_ct = 0;

for ( k=1; k <= nvars; k++)
       {
       if ( var_ratios[k] == 0)
          {
          a_ct = a_ct + 1;
          group_a[k] = pos[k];
          rank_a[a_ct] = k;
          }
       if ( ( var_ratios[k] > 0) && ( var_ratios[k] < 10000000 ))
          {
          b_ct = b_ct + 1;
          group_b[k] = var_ratios[k];
          rank_b[b_ct] = k;
          }
       if ( var_ratios[k] >= 10000000 )
          {
          c_ct = c_ct + 1;
          group_c[k] = neg[k];
          rank_c[c_ct] = k;
          }
       }
/*  Sorting for the three groups                            */
if( a_ct > 1)
       indexx (a_ct, group_a, rank_a);
if( b_ct > 1)
       indexx (b_ct, group_b, rank_b);
if( c_ct > 1)
       indexx (c_ct, group_c, rank_c);
```

```
/* Step 9. Put rank of three groups into combined rank. Rank_a is put */
/* in decreasing order and the rest are put in increasing order       */
a = b = c = 1;
for( k=a_ct; k >=1; k--)
      {
      rank[k] = rank_a[a];
      a = a + 1;
      }
for( k=a_ct+1; k <= b_ct + a_ct; k++)
      {
      rank[k] = rank_b[b];
      b = b + 1;
      }
for( k=a_ct + b_ct + 1; k <= a_ct + c_ct + b_ct; k++)
      {
      rank[k] = rank_c[c];
      c = c + 1;
      }

/* Step 10.          Get initial z_old and z_new value.           */
for( i=1; i<= nterms; i++)
      z_old = z_old + terms[i][0];

z_new = z_old;

k = 1;
for( m=1; m <= nterms; m++)
      {
      if( terms[m][rank[k]] != 0 )
         {
         z_new = z_new - terms[m][0];
         sol[rank[k]] = 0;
         terms[m][0] = 0;
         }
      }


/* Steps 11, 12, 13. Find the minimum value of the objective equation. */
while(( z_old >= z_new ) && ( k <= nvars))
      {
      sol[rank[k]] = 0;
      k = k + 1;
      z_old = z_new;
      for( m=1; m <= nterms; m++ )
         {
         if( terms[m][rank[k]] != 0 )
            {
            z_new = z_new - terms[m][0];
```

```
                terms[m][0] = 0;
                }
            }
        }
sol[rank[k]] = 1;
z_old = z_old * max;

/* Step 14.  Print minimum z value and solution set                    */
printf(" \n\n\n\n\n\n\n\n\n\n\n\nFor Equation:\n\n");
if( max == -1)
        printf("Maximize  ");
else printf("Minimize  ");
for(i=1; i <= nterms; i++)
        {
        if(output_eq[i][0] < 0)
            printf("  %.0f ",output_eq[i][0]);

        else printf("  +%.0f ",output_eq[i][0]);

        for(j=1; j <= nvars; j++)
            {
            if( output_eq[i][j] == 1)
                printf("X%d",j);
            }
        }
printf("\n\n\nPress 1 and enter to see solution\n");
scanf( "%d", &answer);
printf("\n\n\The solution is %.3f.\n\n",z_old);
for( i=1; i <= nvars; i++ )
        printf("Variable %d is %.1f.\n", i, sol[i]);

}
```

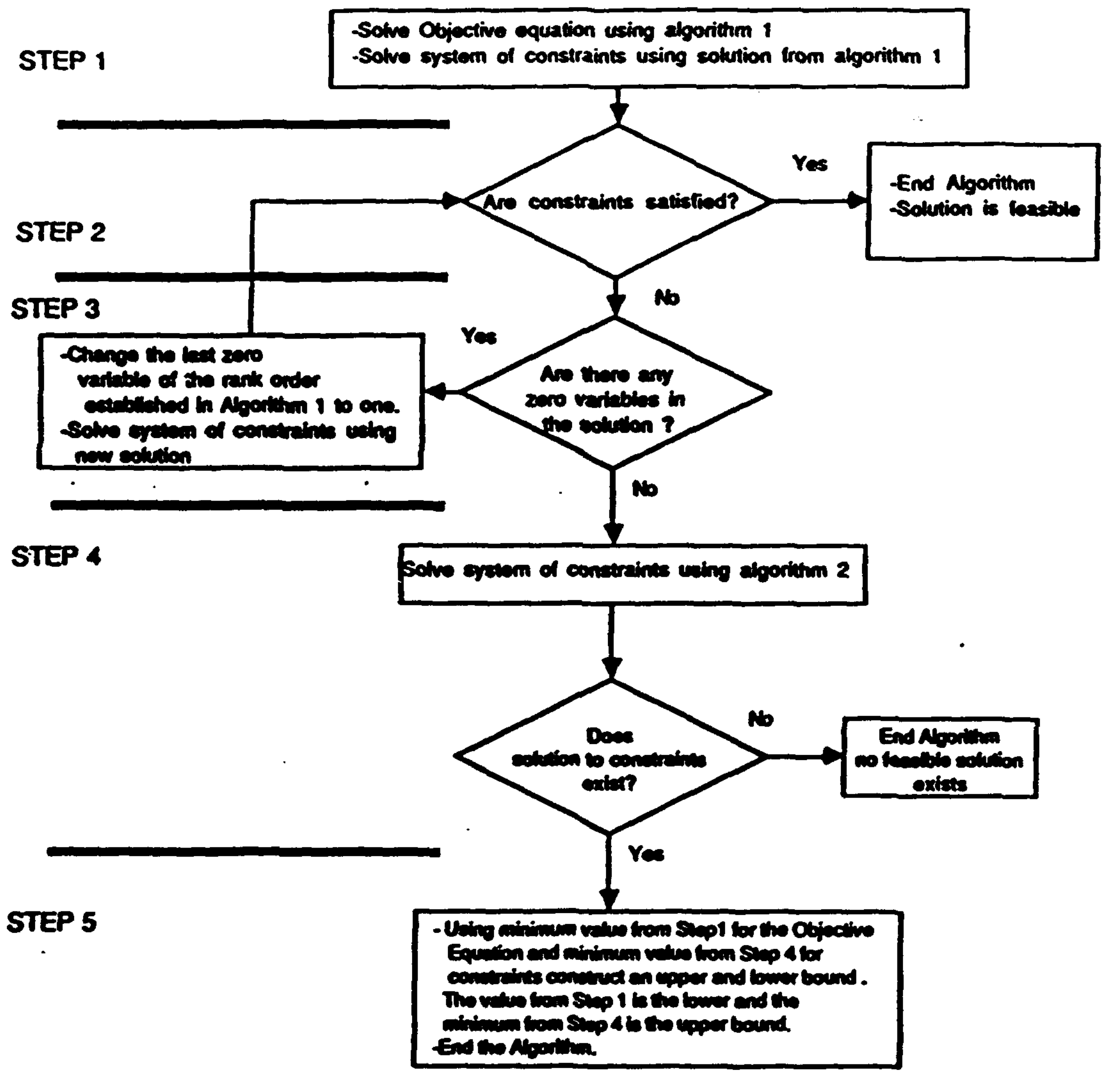
```
/* Receives number of entries of array to be sorted, and array.      */
/* sends sorted index with original array unchanged.                 */

void indexx(n,arrin,indx)
int n,indx[];
float arrin[];
{
      int l,j,ir,indxt,i;
      float q;


      l=(n >> 1) + 1;
      ir=n;
      for (;;)
         {
         if (l > 1)
            q=arrin[(indxt=indx[--l])];
         else
            {
            q=arrin[(indxt=indx[ir])];
            indx[ir]=indx[1];
            if (--Ir == 1)
               {
               indx[1]=indxt;
               return;
               }
            }
         i=l;
         j=l << 1;
         while (j <= Ir)
            {
            if (j < Ir && arrin[indx[j]] < arrin[indx[j+1]])
               j++;
            if (q < arrin[indx[j]])
               {
               indx[i]=indx[j];
               j += (i=j);
               }
            else j=ir+1;
            }
         indx[i]=indxt;
      }
}
```

```
/***********************************************************************
Allocate and free multidimensional arrays
************************************************************************
Author:        Dave Hale, Colorado School of Mines, 12/31/89
Modified by:   Jim Watson, Colorado School of Mines, 7 July 1990
***********************************************************************/

/* allocate a 1-d array        */
void *alloc1  (size_t n1,size_t size)
{
       void *p;

       if ((p=malloc(n1*size)) == NULL)
          return NULL;
       return p;
}

/* free a 1-d array        */
void free1 (void *p)
{
       free (p);
}

/* allocate a 2-d array        */

void **alloc2 (size_t n1, size_t n2, size_t size)
{
       size_t i2;
       void **p;

       if ((p=(void**)malloc(n2*sizeof(void*)))==NULL)
          return NULL;
       if ((p[0]=(void*)malloc(n2*n1*size))==NULL)
          {
          free (p);
          return NULL;
          }
       for(i2=0; i2 < n2; i2++)
          p[i2] = (char*)p[0] + size*n1*i2;
       return p;
}

/*  free a 2-d array    */
void free2  (void **p)
{
       free (p[0]);
       free (p);
}
```

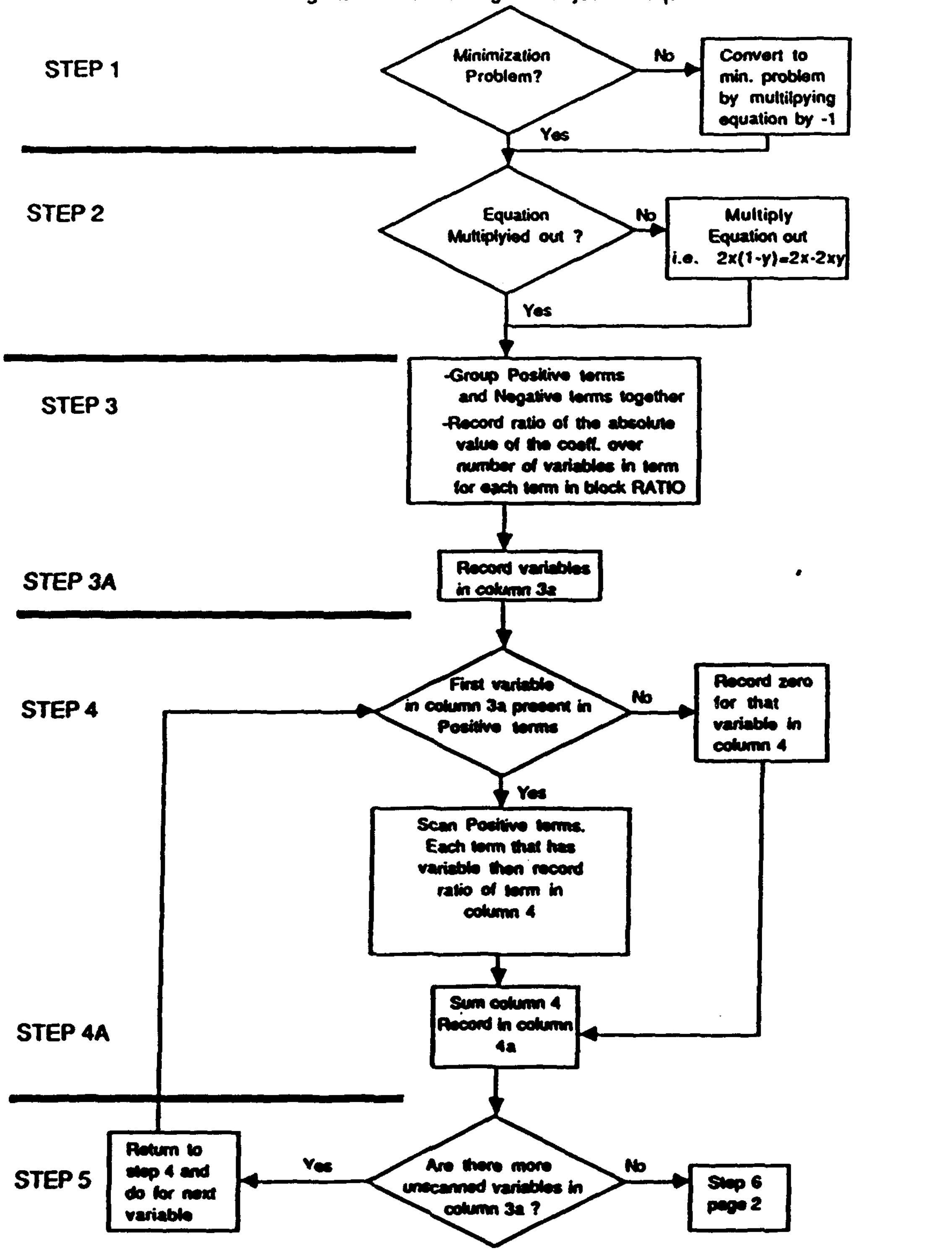
APPENDIX C

FLOWCHARTS

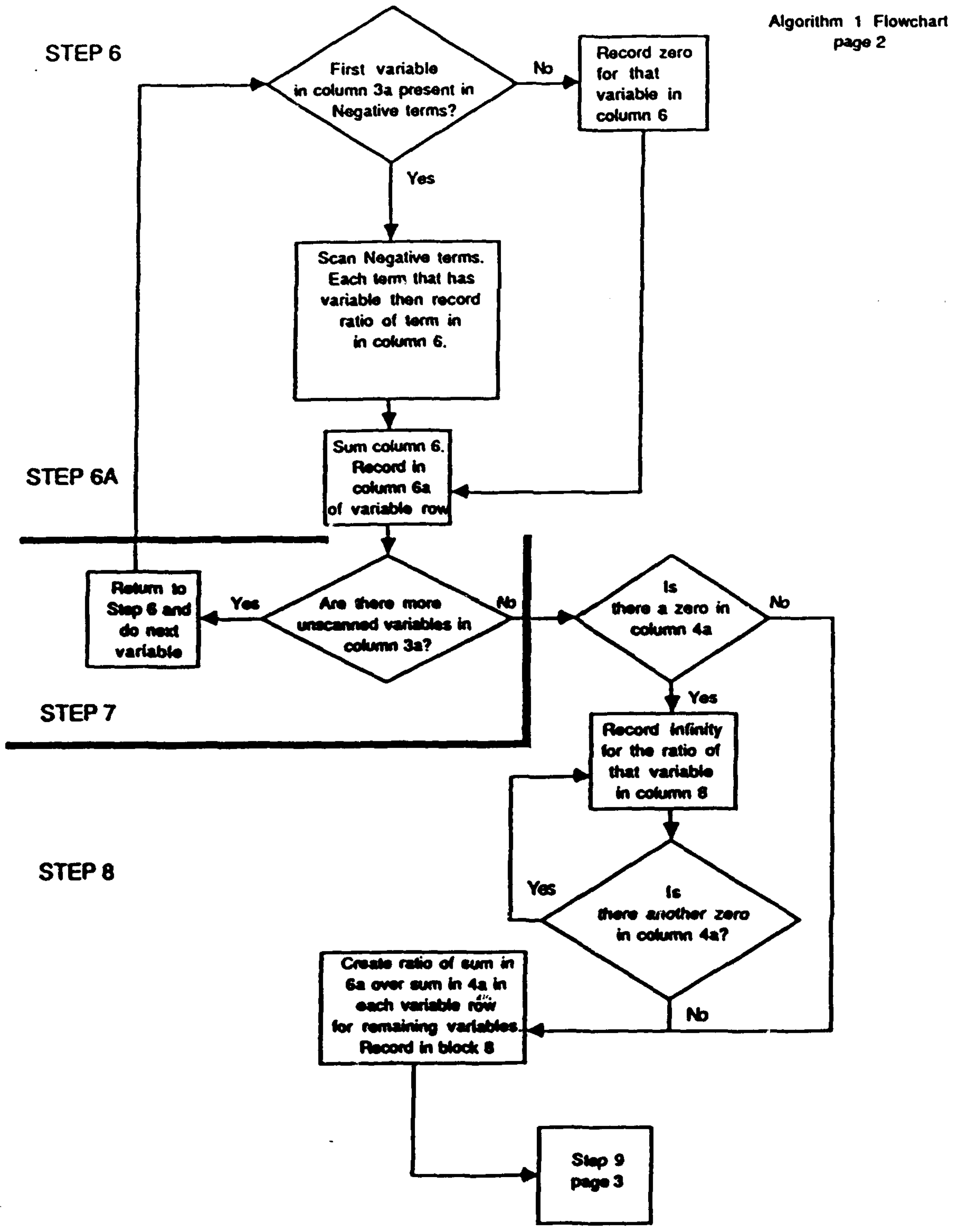
Flowchart for the Overall Algorithm to solve system of equations with an Objective equation and constraints


STEP 1
-Solve Objective equation using algorithm 1
-Solve system of constraints using solution from algorithm 1
STEP 2
Are constraints satisfied?
Yes
-End Algorithm
-Solution is feasible
No
STEP 3
Yes
Are there any zero variables in the solution ?
-Change the last zero variable of the rank order established in Algorithm 1 to one.
-Solve system of constraints using new solution
No
STEP 4
Solve system of constraints using algorithm 2
Does solution to constraints exist?
No
End Algorithm no feasible solution exists
Yes
STEP 5
- Using minimum value from Step1 for the Objective Equation and minimum value from Step 4 for constraints construct an upper and lower bound . The value from Step 1 is the lower and the minimum from Step 4 is the upper bound.
-End the Algorithm.

Algorithm 1 for solving the Objective Equation

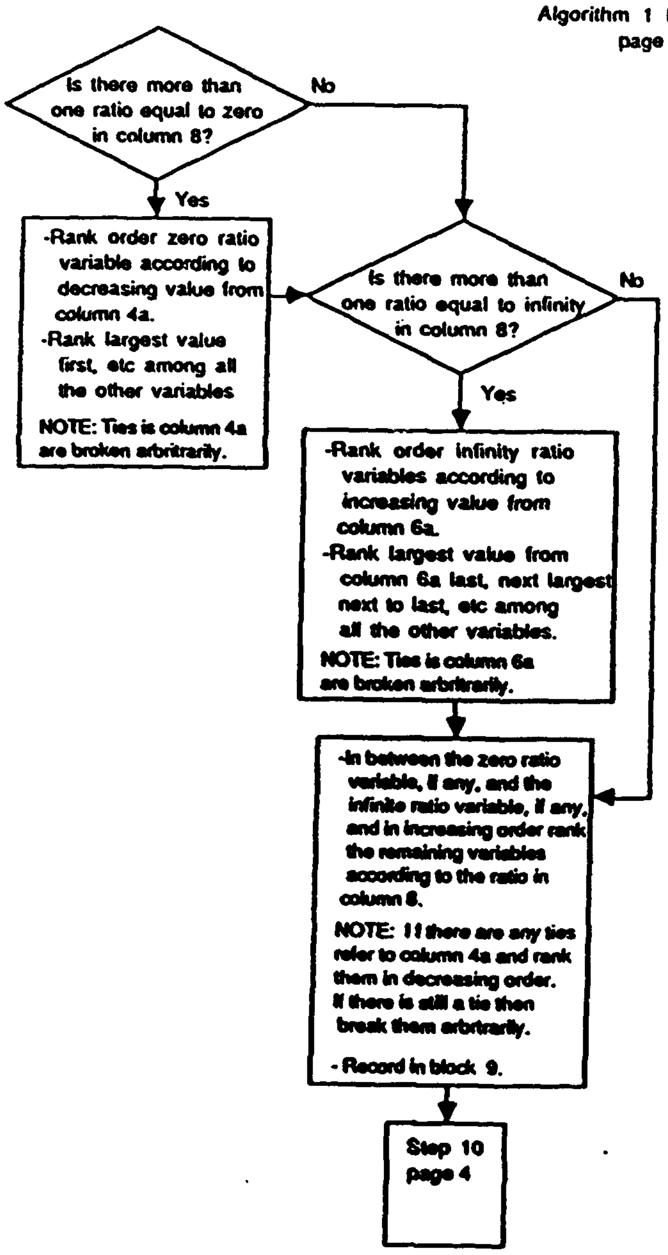

page 1
STEP 1
Minimization Problem?
No
Convert to min. problem by multilpying equation by -1
Yes
STEP 2
Equation Multiplyied out ?
No
Multiply Equation out i.e. 2x(1-y)=2x-2xy
Yes
STEP 3
-Group Positive terms and Negative terms together
-Record ratio of the absolute value of the coeff. over number of variables in term for each term in block RATIO
STEP 3A
Record variables in column 3a
STEP 4
First variable in column 3a present in Positive terms
No
Record zero for that variable in column 4
Yes
Scan Positive terms. Each term that has variable then record ratio of term in column 4
STEP 4A
Sum column 4 Record in column 4a
STEP 5
Return to step 4 and do for next variable
Yes
Are there more unscanned variables in column 3a ?
No
Step 6 page 2

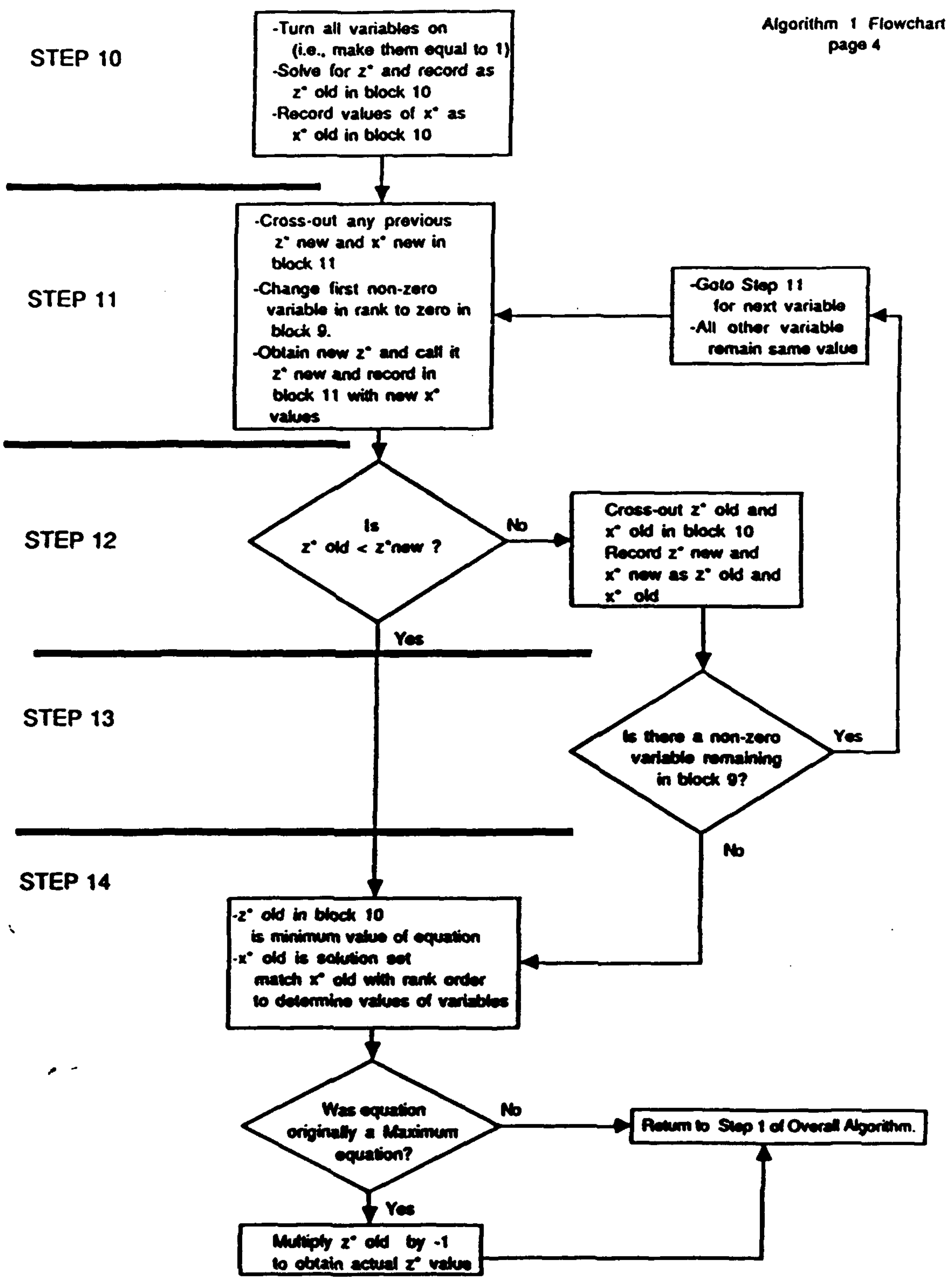
Algorithm 1 Flowchart
page 2


STEP 6
First variable in column 3a present in Negative terms?
No
Record zero for that variable in column 6
Yes
Scan Negative terms. Each term that has variable then record ratio of term in in column 6.
STEP 6A
Sum column 6. Record in column 6a of variable row
Return to Step 6 and do next variable
Yes
Are there more unscanned variables in column 3a?
No
STEP 7
Is there a zero in column 4a
No
Yes
Record infinity for the ratio of that variable in column 8
STEP 8
Yes
Is there another zero in column 4a?
No
Create ratio of sum in 6a over sum in 4a in each variable row for remaining variables. Record in block 8
Step 9 page 3

STEP 9

Algorithm 1 Flowchart
page 3


Is there more than one ratio equal to zero in column 8?
No
Yes
-Rank order zero ratio variable according to decreasing value from column 4a.
-Rank largest value first, etc among all the other variables
NOTE: Ties is column 4a are broken arbritrarily.
Is there more than one ratio equal to infinity in column 8?
No
Yes
-Rank order infinity ratio variables according to increasing value from column 6a.
-Rank largest value from column 6a last, next largest next to last, etc among all the other variables.
NOTE: Ties is column 6a are broken arbritrarily.
-In between the zero ratio variable, if any, and the infinite ratio variable, if any, and in increasing order rank the remaining variables according to the ratio in column 8.
NOTE: If there are any ties refer to column 4a and rank them in decreasing order. If there is still a tie then break them arbrtrarily.
- Record in block 9.
Step 10 page 4


Algorithm 1 Flowchart
page 4
STEP 10
-Turn all variables on (i.e., make them equal to 1)
-Solve for z* and record as z* old in block 10
-Record values of x* as x* old in block 10
STEP 11
-Cross-out any previous z* new and x* new in block 11
-Change first non-zero variable in rank to zero in block 9.
-Obtain new z* and call it z* new and record in block 11 with new x* values
-Goto Step 11 for next variable
-All other variable remain same value
STEP 12
Is z* old < z*new ?
No
Yes
Cross-out z* old and x* old in block 10 Record z* new and x* new as z* old and x* old
STEP 13
Is there a non-zero variable remaining in block 9?
Yes
No
STEP 14
-z* old in block 10 is minimum value of equation
-x* old is solution set match x* old with rank order to determine values of variables
Was equation originally a Maximum equation?
No
Yes
Return to Step 1 of Overall Algorithm.
Multiply z* old by -1 to obtain actual z* value

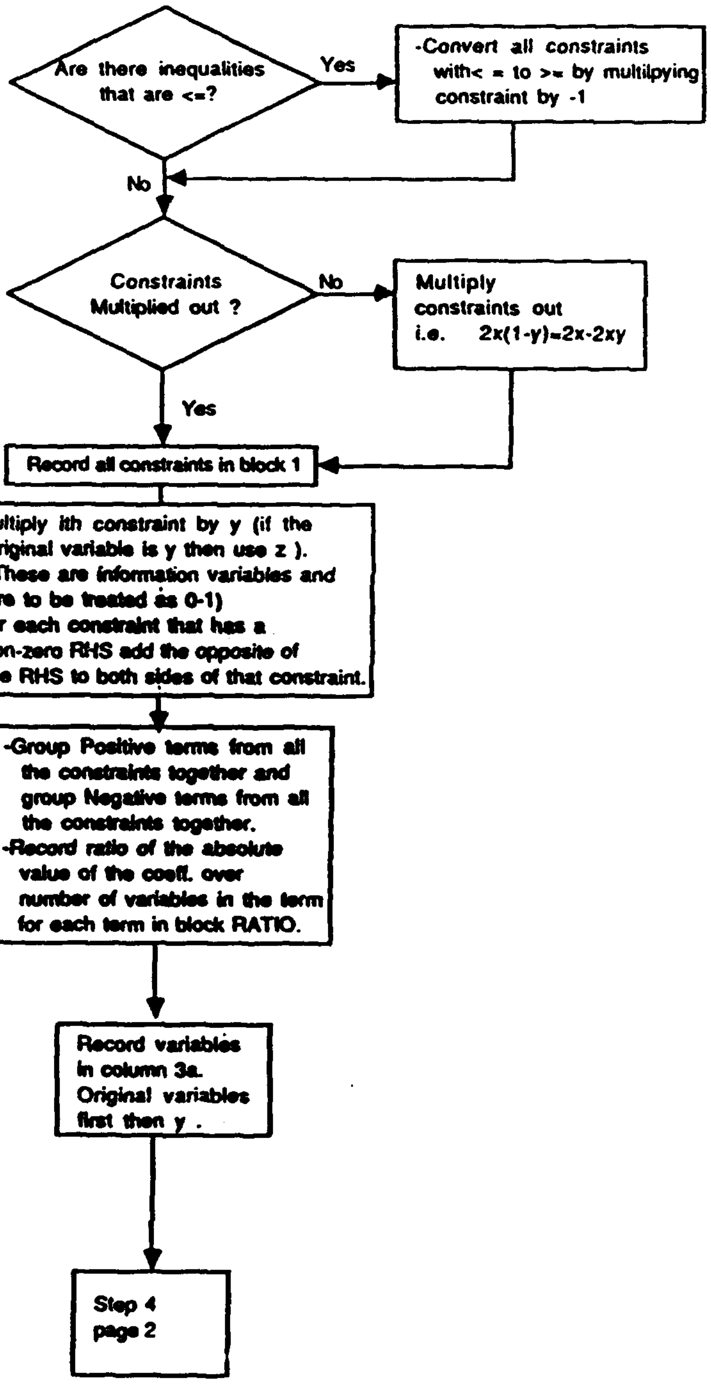
Algorithm 2 for solving the system of constraints page 1

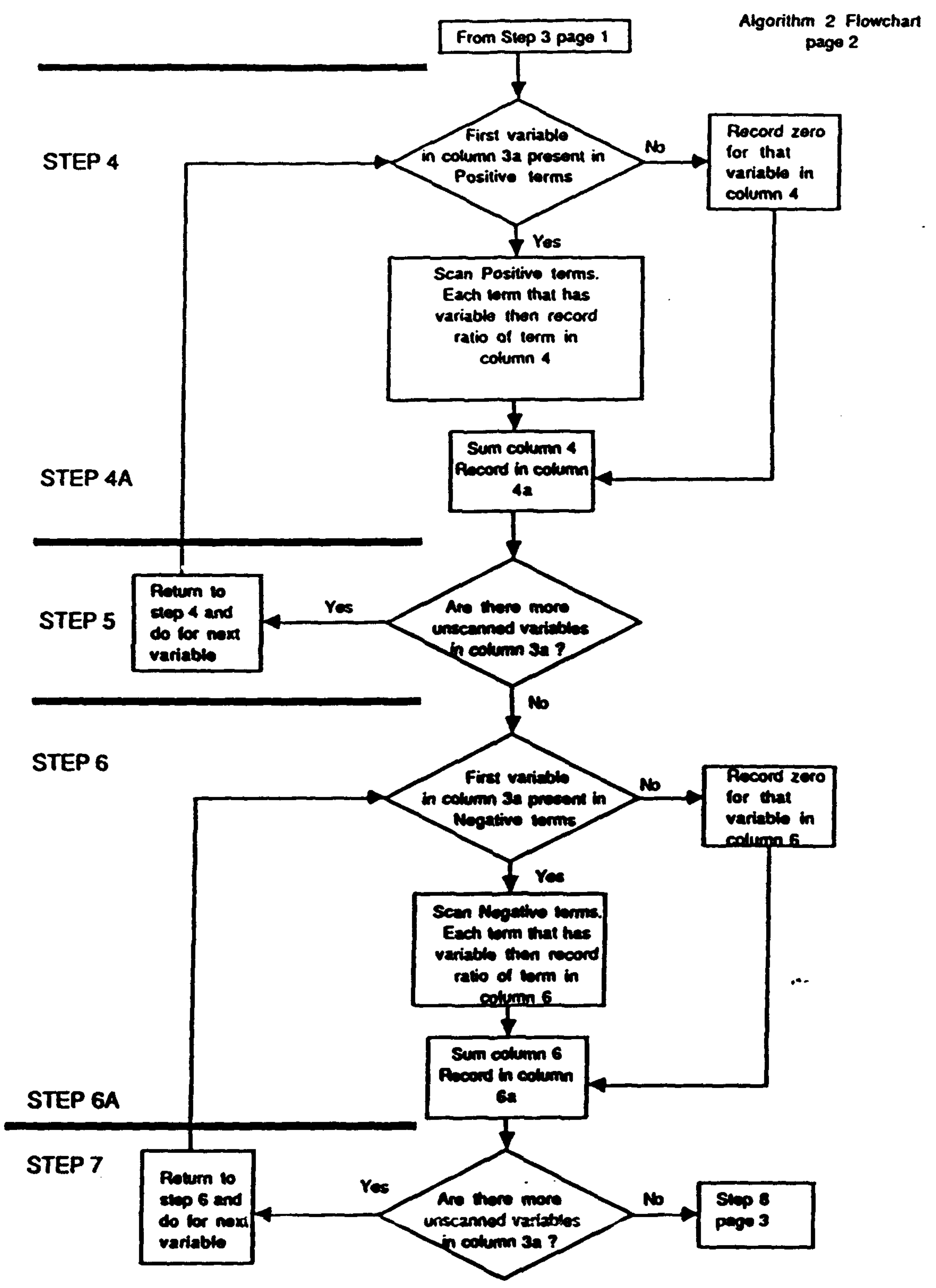
STEP 1

STEP 2

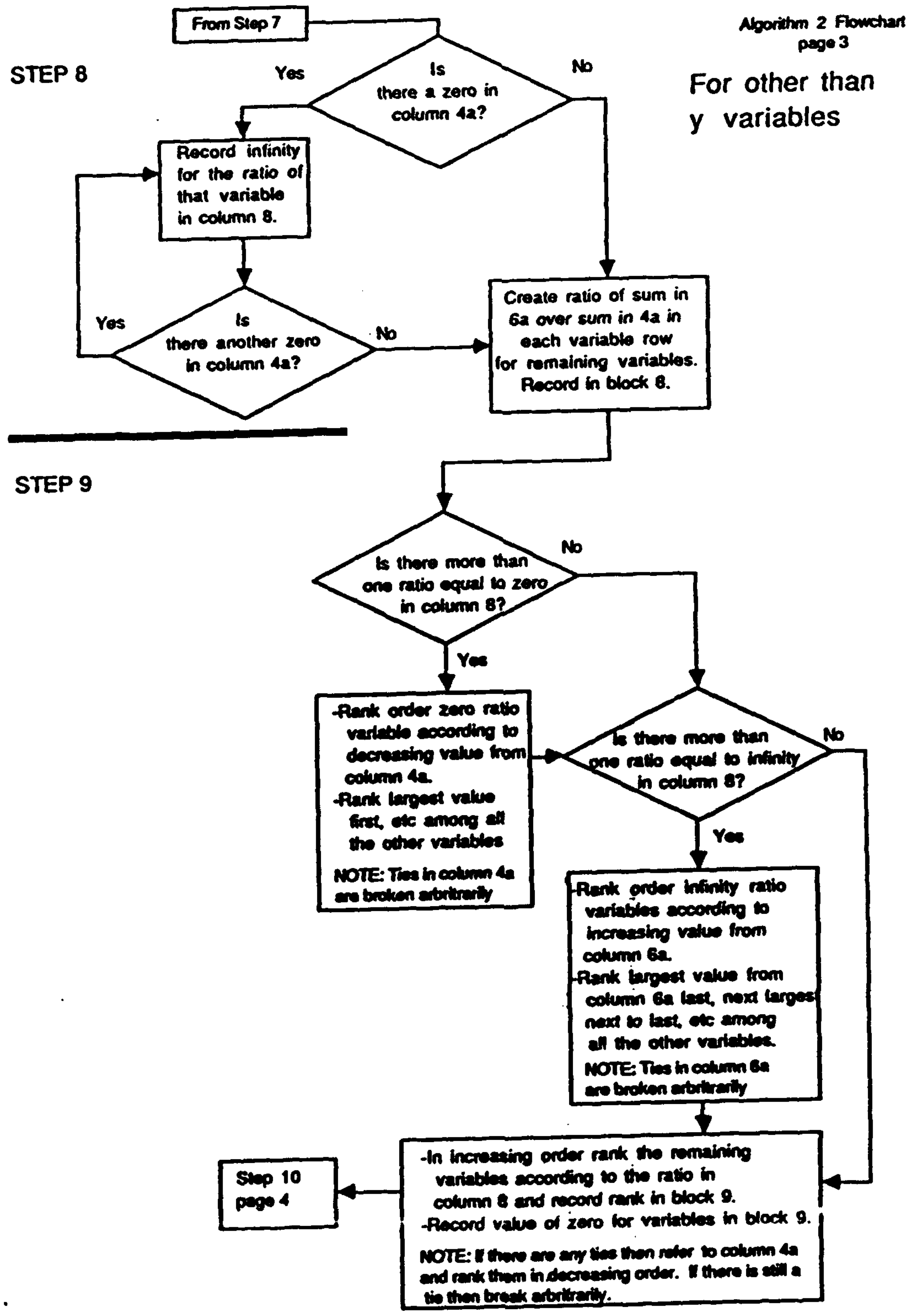
STEP 3

STEP 3A


Are there inequalities that are <=?
Yes
-Convert all constraints with< = to >= by multilpying constraint by -1
No
Constraints Multiplied out ?
No
Multiply constraints out i.e. 2x(1-y)=2x-2xy
Yes
Record all constraints in block 1
-Multiply ith constraint by y (if the original variable is y then use z). (These are information variables and are to be treated as 0-1)
-For each constraint that has a non-zero RHS add the opposite of the RHS to both sides of that constraint.
-Group Positive terms from all the constraints together and group Negative terms from all the constraints together.
-Record ratio of the absolute value of the coeff. over number of variables in the term for each term in block RATIO.
Record variables in column 3a. Original variables first then y .
Step 4 page 2



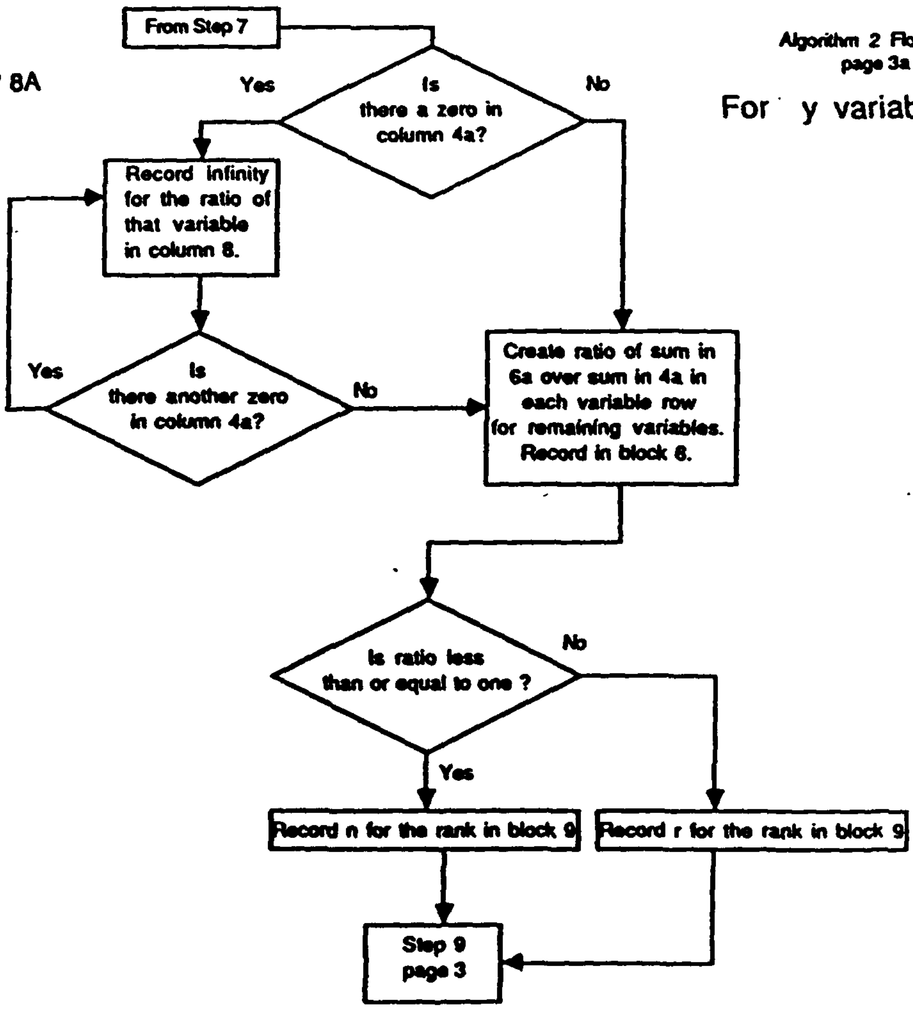
Algorithm 2 Flowchart page 2
From Step 3 page 1
STEP 4
First variable in column 3a present in Positive terms
No
Record zero for that variable in column 4
Yes
Scan Positive terms. Each term that has variable then record ratio of term in column 4
STEP 4A
Sum column 4 Record in column 4a
STEP 5
Are there more unscanned variables in column 3a ?
Yes
Return to step 4 and do for next variable
No
STEP 6
First variable in column 3a present in Negative terms
No
Record zero for that variable in column 6
Yes
Scan Negative terms. Each term that has variable then record ratio of term in column 6
STEP 6A
Sum column 6 Record in column 6a
STEP 7
Are there more unscanned variables in column 3a ?
Yes
Return to step 6 and do for next variable
No
Step 8 page 3


Algorithm 2 Flowchart
page 3
For other than y variables
STEP 8
From Step 7
Is there a zero in column 4a?
Yes
No
Record infinity for the ratio of that variable in column 8.
Is there another zero in column 4a?
Yes
No
Create ratio of sum in 6a over sum in 4a in each variable row for remaining variables. Record in block 8.
STEP 9
Is there more than one ratio equal to zero in column 8?
No
Yes
-Rank order zero ratio variable according to decreasing value from column 4a.
-Rank largest value first, etc among all the other variables
NOTE: Ties in column 4a are broken arbitrarily
Is there more than one ratio equal to infinity in column 8?
No
Yes
-Rank order infinity ratio variables according to increasing value from column 6a.
-Rank largest value from column 6a last, next largest next to last, etc among all the other variables.
NOTE: Ties in column 6a are broken arbitrarily
-In increasing order rank the remaining variables according to the ratio in column 8 and record rank in block 9.
-Record value of zero for variables in block 9.
NOTE: If there are any ties then refer to column 4a and rank them in decreasing order. If there is still a tie then break arbitrarily.
Step 10 page 4

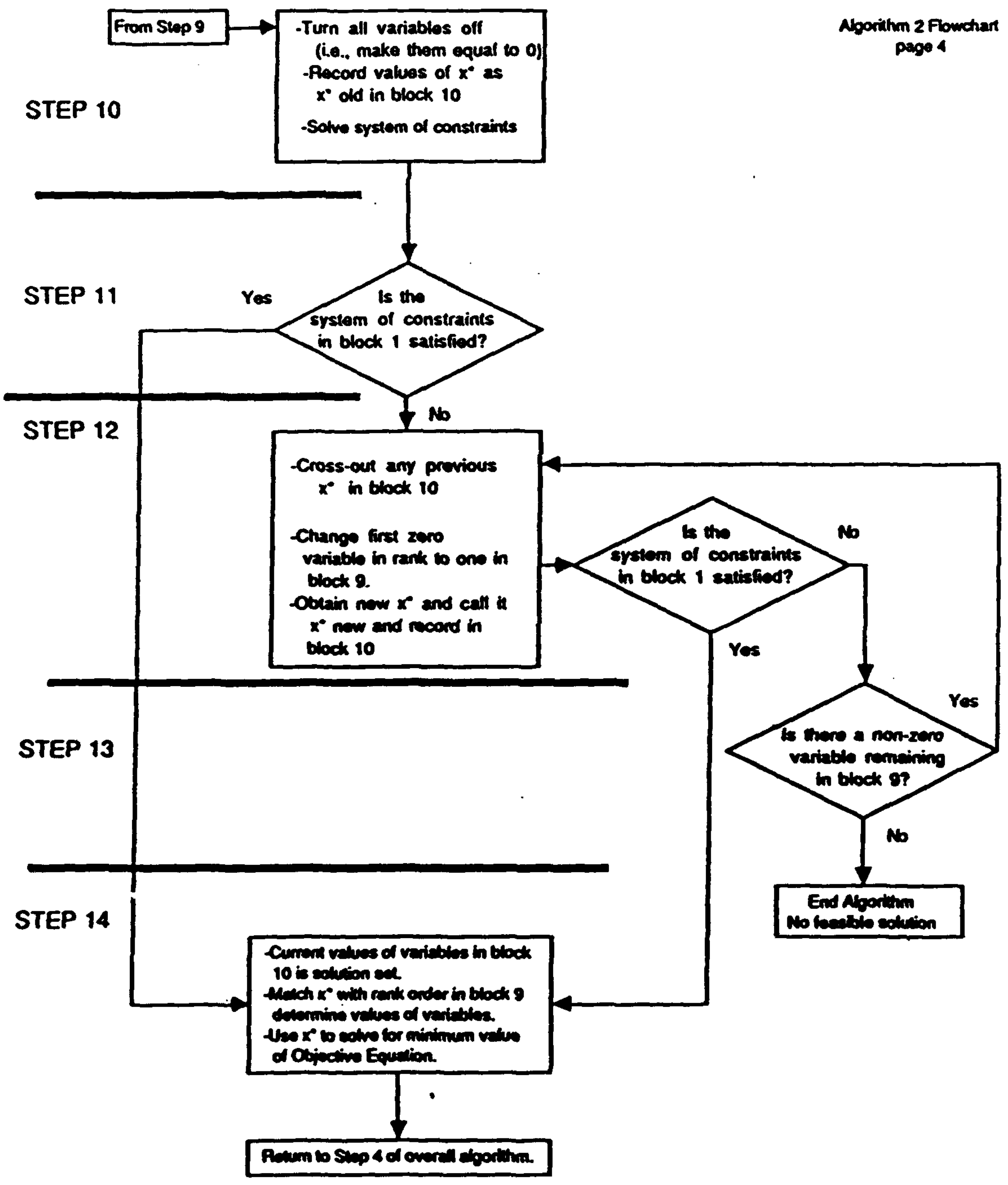
STEP 8A

Algorithm 2 Flowchart
page 3a

For y variables



From Step 7
Is there a zero in column 4a?
Yes
No
Record infinity for the ratio of that variable in column 8.
Is there another zero in column 4a?
Yes
No
Create ratio of sum in 6a over sum in 4a in each variable row for remaining variables. Record in block 8.
Is ratio less than or equal to one ?
No
Yes
Record n for the rank in block 9
Record r for the rank in block 9
Step 9 page 3


Algorithm 2 Flowchart
page 4
From Step 9
STEP 10
-Turn all variables off (i.e., make them equal to 0)
-Record values of x* as x* old in block 10
-Solve system of constraints
STEP 11
Is the system of constraints in block 1 satisfied?
Yes
No
STEP 12
-Cross-out any previous x* in block 10
-Change first zero variable in rank to one in block 9.
-Obtain new x* and call it x* new and record in block 10
Is the system of constraints in block 1 satisfied?
No
Yes
STEP 13
Is there a non-zero variable remaining in block 9?
Yes
No
End Algorithm No feasible solution
STEP 14
-Current values of variables in block 10 is solution set.
-Match x* with rank order in block 9 determine values of variables.
-Use x* to solve for minimum value of Objective Equation.
Return to Step 4 of overall algorithm.

APPENDIX D

WORKSHEETS

1 Minimum Problem

2

3 Group Terms

POS. RATIO	NEG. RATIO

3A Variables	(+) 4	4A	(-) 6	6A	$\frac{(-)}{(+)}$ 8

9

Rank	1	2	3	4	5	6	7	8	9	10	11
variable											
Value											

11

z^* new	x^* new	z^* new	x^* new

10

z^* old	x^* old

14 Solution $z^* =$ $x^* =$

WORKSHEET FOR ALGORITHM 1 (objective equation)BY KEVIN J. LOY

1	
2	

3 Group Terms	
POS. RATIO	NEG. RATIO

3A	(+) 4	4A	(-) 6	6A	8	9 Rank	Value

10 x^*				

14 Solution $x^* =$	$z^* =$

WORKSHEET FOR ALGORITHM 2 (constraints) BY KEVIN J. LOY 18 Oct 90

APPENDIX E
CONDENSED VERSION 0F OVERALL ALGORITHM.

Overall Algorithm for solving System of Pseudo-Boolean Equation

This algorithm is designed for 0-1 variable system of equations. The system of equations usually consists of an objective equation and one or more constraints. The author does not claim optimality.

Step 1

Solve for the minimum value and the solution set of the Objective Equation (OE) using Algorithm 1. Take the solution set of the OE and apply it to the system of constraints (SC).

Step 2

Are the SC satisfied? If yes then end the algorithm because the solution set is feasible. If not then go to Step 3.

Step 3

Are there any zero variables in the solution set? If yes then change the last zero variable in the solution set to one and return to Step 2. If no then go to Step 4.

Step 4

Solve for the minimum value and solution set of the SC using Algorithm 2. If the solution set exists the go to Step 5. If it does not exist then stop algorithm because solution does not exist.

Step 5

At this point the algorithm can only bound the actual value of the OE. Using the minimum value from Step 1 and the minimum value from Step 4 create a bound with a lower bound from Step 1 and an upper bound from Step 4. End the algorithm.

ALGORITHM 1 for Objective Equation

This algorithm is applicable to all 0,1 variable unconstrained problems. Use the worksheet provided to record information. The number of each step corresponds to the number of the block on the worksheet.

Step 1
Is the equation a minimize equation? If no then multiply the problem by -1.
Record the minimum equation in block 1

Step 2
multiply the equation out. Example:
Minimize $-2x_1(1-x_2)+x_3(1-x_4)=-2x_1+2x_1x_2+x_3-x_3x_4$
Record this step in block 2.

Step 3
Record the positive terms of the equation in the block marked POS.
Record the **absolute** value of the negative terms in the block marked NEG
In the block marked RATIO record the ratio of the absolute value of the coefficient of each term over the number of variables in that term.

Step 3a--Record each variable in column 3a putting one variable in each row.

Step 4
Scan each term in the positive block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio below the term in column 4 for the first variable.

NOTE: If the first variable is not present in any of the positive terms then record 0 for that variable in column 4.

Step 4a--Sum the recorded answers in column 4 for the first variable and record that value in column 4a of that variable row.

Step 5
Repeat step 4 and 4a for each variable in column 3a. After this step you should have a real number recorded in column 4a for each variable in column 3a.

Step 6
Scan each term in the negative block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio below the term in column 6 of that variable row.

NOTE: If the first variable is not present in any of the negative terms then record 0 for that variable in column 6.

Step 6a--Sum the recorded answers in column 6 for the first variable and record that value in column 6a of that variable row.

Step 7
Repeat step 6 and 6a for each variable in column 3a. After this step you should have a real number recorded in column 6a for each variable in column 3a.

Step 8
Create a ratio for each variable of the number recorded in column 6a over the number recorded in column 4a. Is there a zero in column 4a? If yes, then record infinity (i.e., ∞) for the ratio of that variable in column 8. Do this for all variables that have a zero in column 4a.

Step 9
In increasing order rank each variable according to the ratio created in column 8. The variable with the smallest ratio is in column 1 of block 9, etc.
Record one for the value of each variable in the value row of block 9.

NOTE: If more than one ratio equal to zero then rank those variables first among the other variables. Then rank them according to the decreasing values from column 4a. If more than one ratio equal to infinity then rank them last among the other variables. Then rank them according the increasing value from column 6a.

NOTE: If there are ties of the non-zero and non-infinity ratios in column 8 then refer to column 4a and rank them in decreasing order.
NOTE: If there are ties in either 6a or 4a then break them arbitrarily.

Step 10
Using the equation from block 2 solve for a z* value where all variables are equal to 1. Call this value z* old and record z* old in block 10.
Record the current value of the variables in block 10 as x* old in vector form (i.e., (0,1,1,1)) in the rank order established in Step 9.

Step 11
Cross out any previous z* new and x* new in block 11.
Change the first non-zero variable in the rank order in block 9 to 0 by crossing out the 1 below the variable in block 9 and record 0.
Obtain a new z* value, with the new variable values, from the equation in block 2.
Call this value z* new and record z* new in block 11.
Call the new variable values x* new.
Record x* new as a vector (i.e., (0,1,1,1)) in the rank order established in Step 9.
Record x* new in the corresponding z* new row of block 11.

Step 12
If the z* old is less than the z* new then stop and Goto step 14.
Else, cross out z* old and x* old in block 10 and record z* new and x* new in block 10 as z* old and x* old.
Goto step 13.

Step 13
Is there a non-zero variable remaining in block 9? If yes, then repeat step 11 for the next non-zero variable. If no, then go to Step 14.

Step 14
z^* old in block 10 is the minimum value of the equation.
The current values of the variables in block 10, the x^* old column, is the solution set.
Match x^* old with the rank order of the variables in block 9 to determine the values of the variables.
Record z^* old and x^* old in block 14.
End the algorithm.

NOTE: If you had to convert the equation to a minimum equation then you must multiple the z^* value by -1 to obtain the actual value of the equation.

ALGORITHM 2 for constraints

This algorithm is applicable to all 0,1 variable constrained problems. Use the worksheet provided to record information. The number of each step corresponds to the number of the block on the worksheet.

Step 1
Are all constraints $\geq$? If no then multiply all constraints that are $\leq$ by -1.
Multiply the constraints out. Example:

$$-2x_1(1-x_2)+x_3(1-x_4) \quad \geq 3 \quad \text{becomes} \quad -2x_1+2x_1x_2+x_3-x_3x_4 \quad \geq 3$$

Record all constraints after this step in block 1.

Step 2
multiply the i-th constraint by y_i if y_i is the original variable then use z_i)and add the opposite of the right hand side (RHS) to both sides of the each constraint.
Record in block 2. This y_i will be used to determine how tight is the i-th constraint. It will be treated as a normal variable but will not be solved for.

Step 3
Record positive terms of all constraints in the block marked POS.
Record the **absolute** value of the negative terms of all constraints in the block marked NEG.
In the block marked RATIO record the ratio of the absolute value of the coefficient of each term over the number of variables, bo[illegible] [illegible]riginal and y_i, in that term.

Step 3a
Record each variable in column 3a putting one variable in each row.
Record all original variables first then the y_i.

Step 4
Scan each term in the positive block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio, below the term, in column 4 for the first variable.

NOTE: If the first variable is not present in any of the positive terms then record 0 for that variable in column 4.
NOTE: Do this for the y variables also.
Step 4a
Sum the recorded answers in column 4 for the first variable and record that value in column 4a of that variable row.

Step 5
Repeat step 4 and 4a for each variable in column 3a. After this step you should have a real number recorded in column 4a for each variable in column 3a.

Step 6
Scan each term in the negative block group looking for the first variable in column 3a. Each time the first variable is present in a term record the ratio below the term in column 6 of that variable row.

NOTE: If the first variable is not present in any of the negative terms then record 0 for that variable in column 6.
NOTE: Do this for the y variables also.
Step 6a--Sum the recorded answers in column 6 for the first variable and record that value in column 6a of that variable row.

Step 7
Repeat step 6 and 6a for each variable in column 3a. After this step you should have a real number recorded in column 6a for each variable in column 3a.

Step 8
For all variables other than the y_i introduced at Step 2.
Create a ratio for each variable of the number recorded in column 6a over the number recorded in column 4a. Is there a zero in column 4a? If yes, then record infinity (i.e., ∞) for the ratio of that variable in column 8. Do this for all variables that have a zero in column 4a.

Step 8a
For all variables y_i introduced at Step 2.
If the ratio for $y_i$is less than one then that constraint in probably not restrictive. Record n (i.e., not restrictive) for the rank in block 9. If the ratio for y_i in greater than one then that constraint is restrictive. Record r (i.e., restrictive) for the rank in block 9. Restrictive is defined to mean most, if not all, of the original variables must be 1. Non-restrictive means the original variables could be either 0 or 1 without violating the constraints.

Step 9
In increasing order rank each variable according to the ratio created in column 8. The variable with the smallest ratio is ranked 1 block 9, etc.
Record zero for the value of each variable in the value column of block 9.

NOTE: If more than one ratio equal to zero then rank those variables first among the other variables. Then rank them according to the decreasing values from column 4a. If more than one ratio equal to infinity then rank them last among the other variables. Then rank them according the increasing value from column 6a.

NOTE: If there are ties of non-zero or non-infinity ratios in column 8 then refer to column 4a and rank them in decreasing order.
NOTE: If there are ties in either 6a or 4a then break them arbitrarily.

<u>Step 10</u>
Record the value of 0 for the variables in block 10 as x* in vector form (i.e., (0,0,0,0)) in the rank order established in Step 9.
Solve the system of constraints.

<u>Step 11</u>
For each constraint does x* satisfied the constraint in block 1?
If all constraints are satisfied then Goto block 14.
Else Goto Step 12

<u>Step 12</u>
Cross out any previous x* in block 10.
Change the first zero variable in the rank order in block 9 to 1 by crossing out the 0 in the variable row of block 9 and record 1.
Record x* new in block 10 in the order established in Step 9.
Solve the system of constraints.
For each constraint does x* new satisfy the constraints in block 1?
If all constraints are satisfied then Goto block 14.
Else, Goto step 13.

<u>Step 13</u>
Is there a zero variable remaining in block 9? If yes, then repeat step 12. If no, then end algorithm because there is no feasible solution.

<u>Step 14</u>
The current values of the variables in block 10, the x* column, is the solution set.
Match x* with the rank order of the variables in block 9 to determine the values of the variables.
Record x* in block 14.
Use x* to solve for the value of the objective equation and record the value (z*) in block 14.
End Algorithm 2.
Return to Overall Algorithm.

APPENDIX F

OUTPUT FOR NUMBER 29

Appendix G contains screen prompts, input, and output.

```
#-------------------------------------------------------------------#
# This program solves Pseudo-Boolean unconstrained Equations.       #
# It assumes:                                                       #
#    1. the equation is multiplied-out(i.e.,2x1(1-x2)=2x1-2x1x2)   #
#    2. the equation is a minimization or maximization              #
#    3. every term has at least one variable                        #
#    4. no more than 30 terms and 30 variables                      #
# The input that can be changed is coefficients and whether         #
# or not variable Xi is present if 0 or 1 is not entered.           #
#                                                                   #
#              The author is:   Kevin J. Loy                        #
---------------------------------------------------------------------
```

```
If equation is a maximum equation then enter -1. Else, enter 1.
Do not enter anything else but these two numbers.
-1
Input each term one at a time. Input the coefficient first with
the appropriate sign. If the variable is persent in that term
then enter 1 to signify that variable is in that term. Else
enter 0 to signify that the variable is not present.


How many terms are there?
6
How many variables are there?
6
What is the coefficient of term 1?
3
If variable 1 is in term 1 then enter 1
Else enter 0
1
If variable 2 is in term 1 then enter 1
Else enter 0
0
If variable 3 is in term 1 then enter 1
Else enter 0
```

```
0
If variable 4 is in term 1 then enter 1
Else enter 0
0
If variable 5 is in term 1 then enter 1
Else enter 0
0
If variable 6 is in term 1 then enter 1
Else enter 0
0
What is the coefficient of term 2?
-1
If variable 1 is in term 2 then enter 1
Else enter 0
0
If variable 2 is in term 2 then enter 1
Else enter 0
1
If variable 3 is in term 2 then enter 1
Else enter 0
0
If variable 4 is in term 2 then enter 1
Else enter 0
0
If variable 5 is in term 2 then enter 1
Else enter 0
0
If variable 6 is in term 2 then enter 1
Else enter 0
0
What is the coefficient of term 3?
-2
If variable 1 is in term 3 then enter 1
Else enter 0
1
If variable 2 is in term 3 then enter 1
Else enter 0
0
If variable 3 is in term 3 then enter 1
Else enter 0
1
If variable 4 is in term 3 then enter 1
Else enter 0
0
If variable 5 is in term 3 then enter 1
Else enter 0
1
If variable 6 is in term 3 then enter 1
Else enter 0
0
What is the coefficient of term 4?
```

```
2
If variable 1 is in term 4 then enter 1
Else enter 0
0
If variable 2 is in term 4 then enter 1
Else enter 0
1
If variable 3 is in term 4 then enter 1
Else enter 0
0
If variable 4 is in term 4 then enter 1
Else enter 0
0
If variable 5 is in term 4 then enter 1
Else enter 0
0
If variable 6 is in term 4 then enter 1
Else enter 0
1
What is the coefficient of term 5?
-1
If variable 1 is in term 5 then enter 1
Else enter 0
1
If variable 2 is in term 5 then enter 1
Else enter 0
0
If variable 3 is in term 5 then enter 1
Else enter 0
0
If variable 4 is in term 5 then enter 1
Else enter 0
1
If variable 5 is in term 5 then enter 1
Else enter 0
0
If variable 6 is in term 5 then enter 1
Else enter 0
1
What is the coefficient of term 6?
2
If variable 1 is in term 6 then enter 1
Else enter 0
0
If variable 2 is in term 6 then enter 1
Else enter 0
0
If variable 3 is in term 6 then enter 1
Else enter 0
0
```

```
If variable 4 is in term 6 then enter 1
Else enter 0
1
If variable 5 is in term 6 then enter 1
Else enter 0
0
If variable 6 is in term 6 then enter 1
Else enter 0
0
The coefficient of each term in order are 3, -1, -2, 2, -1, 2
If this correct, enter 1
Else, enter 0 and re-enter all coefficients.
1
```

```
For Equation:

Maximize    +3 X1  -1 X2  -2 X1X3X5  +2 X2X6  -1 X1X4X6  +2 X4

Press 1 and enter to see solution
1


The solution is 5.000.

Variable 1 is 1.0.
Variable 2 is 1.0.
Variable 3 is 0.0.
Variable 4 is 1.0.
Variable 5 is 0.0.
Variable 6 is 1.0.
```

APPENDIX G

PSEUDO - CODE OF ALGORITHM 1

Pseudo - code of Algorithm 1

The objective equation can be expressed as

$$z = f(x_1,\dots,x_n) = \sum_{j=1}^{m} T_j \text{ where } T_j = c_j \prod_{i=1}^{n} x_i^{\rho_{ij}},\ \ x_i^1 = x_i \text{ and } x^0 = 1\ ,\ \rho_{ij} = \begin{cases} 1 & \text{if } x_i \text{ present in term j,} \\ 0 & \text{if not present in term j.} \end{cases}$$

For objective equation or function $z = f(x_1, x_2, \dots, x_n)$ with:

x_i = i-th variable, $1 < i < n$, $x_i = 0$ or 1.
n = number of variables
m = number of terms
c_j = coefficient of j-th term, $0 < j < m$.

1. If problem is not a min problem then convert to min problem by multiplying z by -1.

2. Multiply problem out into simplified form (i.e. no parentheses).

3. For every term: $r_j = \frac{|c_j|}{\sum_{i=1}^{n} \rho_{ij}}$. The r_i is the <u>term ratio</u>.

4. For i = 1 to n:

$$0 \le d_i^+ = \sum_{j=1}^{m} r_j \rho_{ij} \psi_j^+, \text{ where } \psi_j^+ = \begin{cases} 1 \text{ if } c_j > 0 \\ 0 \text{ if } c_j < 0 \end{cases}$$

$$0 \le d_i^- = \sum_{j=1}^{m} r_j \rho_{ij} \psi_j^-, \text{ where } \psi_j^- = \begin{cases} 1 \text{ if } c_j < 0 \\ 0 \text{ if } c_j > 0 \end{cases}$$

5. Since d_i^+ and d_i^- will not equal zero at the same time:

$$v_i = \frac{d_i^-}{d_i^+} \quad \text{where} \quad \begin{cases} \text{if } d_i^+ = 0 \text{ then } v_i = \infty \\ \text{if } d_i^- = 0 \text{ then } v_i = 0 \end{cases}.$$ The v_i is the <u>variable ratio</u>.

6. Sort v_a s.t. $v_a < \dots < v_k$ where a is the subscript of the smallest ratio and k is the subscript of the largest ratio. Each variable is associated with a ratio and ranked in the same order as the associated ratio. For example, $x_a, \dots, x_k$ is the ranking with the same subscripts.

7. Let all $x_i = 1$.
Solve for $f(x_1, x_2, \dots x_n) = z^*$ where all $x_i = 1$.

8. For i = subscript of variable with smallest ratio in rank to subscript of variable with largest ratio. Do this if there exists $x_i \neq 0$. If every $x_i = 0$ then go to 9.

```
Let $x_a = 0$ where a is the subscript of first non-zero variable in rank order.
Solve for $f(x_1, \ldots, x_n) = z^{**}$ using current values of $x_i$
    If $z^* < z^{**}$ then stop and go to 9.
    Else $z^* = z^{**}$ and return to 8.
```

9. z^* is solution and current values of x_i is solution set.