1-1-1979

A computerized solution of polyomial geometric programs.

Lewis Craig Chasalow

A COMPUTERIZED SOLUTION OF POSYNOMIAL GEOMETRIC PROGRAMS

by

Lewis Craig Chasalow

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

1979

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science in Industrial Engineering.

May 2, 1979
(date)

Dr. Louis J. Plebani
Professor in Charge

Mr. George E. Kane
Chairman
Department of Industrial Engr.

This thesis is dedicated to my parents and Loren, without who's support I would not have had the strength to see it through.

TABLE OF CONTENTS

Abstract

This Thesis is an investigation of a technique which can be used to solve geometric programming problems. Geometric programming is a method by which a certain class of nonlinear optimization problems may be solved by solving an associated problem with a concave objective function and linear constraints. This convex program is formed by a dual transformation of the original (primal) problem.

Several techniques were investigated which could be used to solve linearly constrained convex programs. A method proposed by Zangwill (3) which uses a generalized Simplex method, was concluded to be well suited to solving such problems.

Several difficulties were encountered with using this Convex Simplex method to directly solve geometric programs. An alogorithm which modified this method to deal specifically with these problems was found to have been proposed in a paper by Beck and Ecker (4). This algorithm was subsequently chosen as the major area of concentration of this thesis.

A computer program was written to solve geometric programming problems using this modified convex simplex

method. Test problems were solved using this computer program and the CDC 6400 computer at Lehigh, in order to examine the operation of this algorithm. Several conclusions were drawn as to how the algorithm could be expected to perform on various classes of problems.

In general, the modified convex simplex method was found to be quite useful for solving high degree of difficulty geometric programs. Further study could lead to generalization of this algorithm to allow solution of an even broader range of problems, specifically signomial problems. Finally, a comparative study of this and other algorithms was proposed to provide a standardized solution technique for all geometric programming problems.

1: Introduction-

1.1: Background-

Geometric programming is a technique developed by Duffin, Peterson, and Zener (1) to solve a particular class of nonlinear mathematical optimization problems. These problems are characterized by posynomial (positive polynomial) objective functions and constraints. A posynomial is any polynomial function which has only positive coefficients. They found that many problems of this form arise naturally in such fields as economics and engineering design. The basic form of these problems is as follows:

$$\text{Minimize } y_0(\underline{x}) = \sum_{t=1}^{T_0} c_{0t} P_t(\underline{x})$$

subject to

$$y_1(\underline{x}) = \sum_{t=1}^{T_1} c_{1t} Q_t(\underline{x}) \leq 1$$

$$\underline{x} > 0$$

where

$$P_t(\underline{x}) = \prod_{n=1}^{N} x_n^{a_{0tn}}; \qquad t = 1, 2, \ldots, T_0$$

$$Q_t(\underline{x}) = \prod_{n=1}^{N} x_n^{a_{1tn}}; \qquad t = 1, 2, \ldots, T_1$$

(2)

where: c_{mt} is the coefficient of the i^{th} term in the m^{th} constraint (must be positive).

x_n is the n^{th} primal variable

a_{mtn} is the exponent of the n^{th} variable, t^{th} term, m^{th} constraint. (m=0 is the objective function)

T_m is the number of terms in the m^{th} constraint.

N is the number of variables.

M is the number of constraints.

This program is refered to as the primal. The method relies on solving the associated dual problem of the form:

$$\text{Maximize } d(w) = \prod_{m=0}^{M} \prod_{t=1}^{T_m} \left(\frac{c_{mt}\omega_{mo}}{\omega_{mt}} \right)^{\omega_{mt}}$$

subject to

$$\sum_{t=1}^{T_o} \omega_{ot} = 1$$

$$\sum_{m=1}^{M} \sum_{t=1}^{T_m} a_{mtn}\omega_{mt} = 0; \qquad n = 1, 2, \ldots, N$$

$$\omega_{mo} = \sum_{t=1}^{T_m} \omega_{mt}; \qquad m = 1, 2, \ldots, M$$

(2)

where: ω_{mt} is the dual variable associated with the t^{th} term in the m^{th} constraint.

The dual variables represent the proportion of the

associated primal term to the value of the equation which it is in, either objective function or constraint.

1.2: Theory Behind the Technique-

The relationship between the objective function of the primal program and the dual objective is entirely analogous to the relationship between the arithmetic and geometric means.(1) It is this relationship which is the basis for much of the theory of geometric programming and which led to the name of the technique. This relationship assures that if a maximum value can be found for the dual program, it must also be a minimum for the primal. In fact, the test of optimality of the dual program is convergence of the dual objective function with the primal. (A more detailed discussion of this topic is included in appendix A)

Each variable in the dual formulation corresponds directly to a term in the primal program, the values of these terms being the proportions mentioned earlier. These proportions are independent of the coefficients of each term. This means that no matter what coefficients are used, the dual variables will give the optimal solution. This can be especially useful if each term has some physical significance and there is some

uncertainty in the coefficients. The geometric programming solution will give useful results which will lead to optimality no matter what uncertainty exists.

1.3: Problem Solving Practicalities-

The method of solving for these dual variables, in many cases, requires merely solving a set of simultaneous linear equations. A look at the dual constraints reveals that they form a set of N+1 equations with T unknowns (T is the total number of primal terms in the problem). If N+1 equals T a unique solution exists which can be solved for explicitly, as long as the equations are nonsingular. If the number of terms in the primal program is greater than one plus the number of variables, there are an infinite number of solutions to these equations.

Because the difficulty in solving these problems increases as T-(N+1) increases, this value is called the degree of difficulty of a problem. It represents the number of dual variables which cannot be explicitly determined from the dual constraints. Many techniques have been proposed to try to solve the problem of high degree of difficulty. However, even with high degrees of difficulty, geometric programming is valuable in that it allows the solution of a highly nonlinear

problem by solving a problem with a concave objective function and linear constraints.(2)

The constraints of the dual formulation are obviously linear, and the objective function is easily shown to be concave. Using the logarithmic form, the objective function becomes the sum of the following terms, $\omega_{mt}\ln(c_{mt}\omega_{mo}/\omega_{mt})$ for $t=1,\ldots,T_m$; $m=0,\ldots,v$. As long as the dual variables are positive (which is a requirement of the method) each of these terms is positive and therefore the sum of these terms will be a concave function.(3)

1.4: Generalized Geometric Programming-

All of the above discussion deals with posynomial problems only. In practice problems may come up which can only be described using signomial (signed polynomial) functions. A signomial is any polynomial, which may have either positive or negative coefficients. There are considerations unique to signomial problems which need not be addressed when dealing with posynomials. For example, because the objective function of the dual problem is no longer concave, the solution of a signomial problem will not necessarily yield a global optimum.(2)

Posynomial problems occur often enough in practice for a method which solves posynomial problems only to

be valuable. Methods have also been proposed which would transform signomial problems into posynomials for solution, thereby eliminating the need for a special algorithm to solve signomials in many cases (7). For these reasons this thesis will be confined to the discussion of posynomial programs.

2: Algorithms to deal with degrees of difficulty-

2.1: Low Degrees of Difficulty-

The first step to take when confronted with degrees of difficulty would be to examine how the problem was formulated. It may be possible to eliminate certain terms from the problem thereby reducing the degree of difficulty(?). Of course, for many problems it isn't possible to eliminate degrees of difficulty. For this reason there must be some method of dealing with degrees of difficulty.

One way to solve this type problem is to solve the dual constraints in terms of the variables represented by the degrees of difficulty. This solution is then substituted into the dual objective function. Finally, this function is optimized either by differentiation or by searching over the range of the unknown variables (?).

2.2 Higher Degrees of Difficulty-

The technique just described will obviously become unwieldy very rapidly for more than two or three degrees of difficulty. Therefore, some other technique must be used. One way to approach this problem is to take advantage of the fact that the dual program is actually a convex programming problem (concave objective function

and linear constraints). Zangwill proposed a method to solve such problems which is based on the Simplex method. The details of this method can be seen in the reference (3).

Beck and Ecker further modified this Convex Simplex method to allow its use for solving geometric programming problems (4). This technique appears promising for solving high degree of difficulty problems and will be the primary topic of the rest of this thesis.

2.3: The Modified Convex Simplex Method-

Two major modifications were needed to make the Convex Simplex method applicable to geometric programming. The first was to handle problems with inactive primal constraints. If an inactive primal constraint exists, all dual variables associated with that constraint will be zro at optimality. If the variables associated with this constraint are allowed to vary individually the technique could cycle infinitely between changes in these variables, causing non-convergence of the method (2).

In the original method only one variable was allowed to change at a time. Beck and Ecker modified the algorithm to allow blocks of variables corresponding to inactive primal constraints to change at once (4). This change

eliminates the possibility of cycling, thereby insuring convergence.

Also, at each iteration a gradient vector composed of the partial derivatives of the dual objective function with respect to each dual variable must be computed. (The equations used to calculate this gradient are given in step 1c of the algorithm) If any dual variables are allowed to become zero, the components of this gradient become difficult to evaluate due to a zero term in the denominator of the gradient equations. The block decrease provides that the gradient components within the block remain constant as the variables approach zero, so that the gradient is well defined even with inactive primal constraints (4).

The second change had to do with the initial basic feasible dual solution. Zangwill required that all nonbasic variables be zero initially (3). Beck and Ecker (4) changed this requirement so that all dual variables must be positive (non-zero and non-negative) initially. This requirement insures that if an initial vector conforming to these conditions can be found, the problem is canonical and a solution exists. It also provides for a gradient vector which is well defined initially, so that no approximations need be made. Now a more detailed

description of this alorithm will be given. (4)

Step 0: Initialization-

Set up the initial tableau using the exponent matrix of the primal program. This tableau can be represented by the matrix equation Ty = b ; where y is the vector of dual variables, b is the right hand side of the dual constraints, and T is the body of the tableau. The components of T are represented by t_{ji} where j is the row and i is the column with each row corresponding to a basic variable and each column with a dual variable. These components initially have the values a_{mtn} from the earlier notation with each row containing the 'a' exponents associated with a given primal variable. One more row is then added which has ones in the columns corresponding to the dual variables associated with the primal objective function, and zeros elsewhere. The initial right hand side has a one is this last row and zeros elsewhere.

The phase I Simplex is then used to pivot in an initial basis. In the phase I Simplex a linear program is solved using the given constraints and an objective function of the sum of the artificial variables. From the solution of this problem a positive initial dual feasible vector can be calculated. If such a vector

can't be found the problem is not canonical and the algorithm is terminated. Otherwise, proceed to step 1.

Step 1:

Step one is performed in five stages which result in the values which are needed to determine the nonbasic variable to change.

a) The first stage is to calculate the sum of the dual variables associated with each primal constraint, Λ_k.

$$\Lambda_k = \sum_{(k)} y_i \qquad ; \; k=1,2,\ldots,M$$

$$\Lambda_o = 1.0$$

(k) represents the dual variables associated with the k^{th} primal constraint, k=0 represents the objective function.

b) Next the components of the objective function associated with each dual variable are calculated as well as the total value of the dual objective function. These will be used to determine the gradient vector components and are calculated as follows:

$$v_i = \left\{ \begin{array}{ll} y_i \ln(c_{mt\ \ m}/y_i) & , \; y_i \neq 0 \\ 0 & , \; y_i = 0 \end{array} \right\} \quad \text{for } i=1,\ldots,T$$

$$V = \sum_{i=1}^{T} v_i$$

This equation actually describes the natural log of the dual objective function, so for comparison with

the primal the value exp(V) must be used.

c) Using the objective function components calculated above, the gradient vector components are now computed. This vector consists of the partial derivatives of the objective function with respect to the current dual variables, and is used to describe the contribution of each dual variable to the objective function, it is calculated as follows:

$$G_i = \begin{cases} v_i/y_i - V & \text{, for i within the objective function} \\ v_i/y_i & \text{, elsewhere} \end{cases}$$

$$\text{for } i = 1,\ldots,T$$

d) Now the relative cost vector is computed. This vector will be used to determine the nonbasic variable to change. It represents the change in the objective function for each unit change in the dual variables, and is calculated as follows:

$$C_i^* = G_i - \sum_{j=1}^{N+1} t_{ji} G_{b_j} \quad , \quad i=1,\ldots,T$$

where G_{b_j} is the gradient component of the j^{th} basic variable, and t_{ji} is as defined earlier.

e) Finally, the optimal changes in the nonbasic variables can be calculated. These values will be used in a later step to determine which nonbasic variable is to change if a block decrease isn't indicated.

They are:

$$C^{\bullet}_{s1} = \max_{i=1,\ldots T} (C^{\bullet}_i)$$

$$C^{\bullet}_{s2} = \min_{i=1,\ldots,T} (C^{\bullet}_i y_i)$$

If $C^{\bullet}_{s1} = C^{\bullet}_{s2} = 0$, the current dual vector is optimal and proceed to step 7 to calculate the optimal primal variables. Otherwise, continue to step 2.

Step 2: Check for possible block decreases-
If there exists a block of variables corresponding to a primal constraint for which all $C^{\bullet}_i$ are less than zero, then that block is a candidate for a possible decrease. If it is feasible to set all dual variables in the block to zero, a block decrease is indicated and proceed to step 5e to determine the direction vector. Otherwise, continue to step 3.

Step 3:

The relative cost vector components represent the increase in the dual objective function for each unit increase in the corresponding dual variable. If one of these costs is negative and the dual variable is positive, that variable can be decreased to bring about an increase in the objective function. The greatest increase in the objective function will result from either an increase in the dual variable with maximum positive relative

cost or a decrease in the variable with the minimum product of relative cost and dual variable. Therefore, the variable to change is determined using C^*_{s1} and C^*_{s2} as follows:

if $C^*_{s1} \geq |C^*_{s2}|$, select $y_s = y_{s1}$ to increase,

if $C^*_{s1} < |C^*_{s2}|$, select $y_s = y_{s2}$ to decrease.

Where: y_{s1} is the dual variable associated with C^*_{s1}, y_{s2} is the variable associated with C^*_{s2}, and y_s is the variable to change.

Now proceed to step 4.

Step 4:

If y_s is greater than zero go to step 5a, otherwise $y_s=0$ and the variable to be increased is in a block which has been set to zero by a previous block decrease. In this case either a block increase is indicated, or the ratios of the dual variables to Λ_s can be adjusted so that a strict increase in the dual objective function is guaranteed.

These ratios were calculated when the block was last decreased to zero using the equation:

$$r_i = y_i/\Lambda_1 \quad \text{where (1) is the block affected and } i \in (1).$$

The amount by which the ratios associated with negative relative costs can be decreased and those

associated with positive relative costs increased, Δ^- and Δ^+ respectively, are calculated as follows:

$$\Delta^+ = \sum_{\delta^+} r_i(1-\exp(C_i^*)) \text{ , where } \delta^+ = [\, i \in (1) \mid C_i^* > 0]$$

$$\Delta^- = \sum_{\delta^-} r_i(\exp(C_i^*)-1) \text{ , where } \delta^- = [\, i \in (1) \mid C_i^* < 0]$$

If Δ^+ is less than or equal to Δ^-, the relative costs can be made all non-positive by increases in r_i for $i \in \delta^+$ and decreases in r_i for $i \in \delta^-$. These increases are the components of Δ^+ and are offset by decreases in r_i for Δ^-. Then proceed to step 1d.

If Δ^+ is greater than Δ^- the increases in the ratios for Δ^+ cannot be offset by decreases in the ratios for Δ^-. In this case the relative costs in the block can all be made non-negative by appropriate decreases in r_i for Δ^- and the direction vector for the block increase is calculated in step 5d.

Step 5:

Step 5 involves calculating the direction vector for an iteration which describes the maximum feasible change in the dual variables. This step is divided into five parts to deal with all possible cases.

a) y_s is to be increased and at least one component of the tableau is column s is positive. In this case the increase in y_s could drive some basic variable.

to zero. The direction vector, d, is determined as follows:

$$d_s = \min_{j=1,\ldots,N+1} (y_{b_j}/t_{js} \; ; \; t_{js} \; 0)$$

$$d_{b_j} = -t_{js} \, d_s \qquad , \; j=1,\ldots,N+1$$

$$d_i = 0 \qquad , \text{ elsewhere.}$$

d_{b_j} is the direction vector component for the j^{th} basic dual variable and d_s is the component for the variable to be increased. Since the move along the direction vector is accomplished by adding some fraction of each direction vector component to the corresponding dual variable, this choice for d_s insures that all dual variables will remain non-negative.

The direction vector components for the basic variables come from the fact that the sum of the dual variables times each term in the row must equal the same value after the move. This is because each row is a constraint which must be satisfied both before and after the move. Therefore, if y_s is increased by d_s, the component of the row product due to y_s is increased by $t_{js}d_s$ for each row, and the basic variable in each row must be reduced by an equal amount to keep each constraint satisfied.

Set θ=1 and go to step 6. θ represents the

maximum feasible move along the direction vector.

b) y_s is to be increased and all components of the tableau in column s are less than or equal to zero. In this case no matter how much y_s is increased, no basic variables will be driven to zero. Since the basic variables are not a concern, d_s could take on any real positive value. For reasons of practicality d_s is set to 1 and $\theta = 100$. The objective function may actually be maximized for a move of greater than 100 times the direction vector, but this will be taken into account by future iterations. The rest of the direction vector is determined in the same manner as for 5a, then go to step 6.

c) y_s is to be decreased. The direction vector is determined similarly to step 5a, except:

$$d_s = \max(-y_s, \max_{j=1,\ldots,N+1}(y_{b_j}/t_{js} \mid t_{js} < 0))$$

In this case the concern is not only that basic variables are not driven negative, but y_s must also be kept positive. The second maximization above insures that the basic variables are never driven negative. If t_{js} is negative for a basic variable, that variable will be decreased by $t_{js}d_s$, therefore for any variable the maximum value that d_s can have is y_j/t_{js} for that

basic variable to remain non-negative. By taking the maximization of these negative values this limitation is at least met for all basic variables.

Then this value is limited by the fact that the variable being decreased must also reamin non-negative. This leads to the form of the equation above which insures that d_s isn't larger than the magnitude of this variable.

Now θ is set to 1 and proceed to step 6.

d) An entire block, previously decreased to zero, is being increased away from zero. The ratios corresponding to the dual variables in the block have been adjusted in step four so that all relative costs are positive within the block. The decrease in each basic variable per unit increase in the block, γ_{b_j}, is calculated as follows:

$$\gamma_{b_j} = - \sum_i t_{ji} r_i \qquad \text{, for } j=1,\ldots,N+1$$

for i within the block affected.

This equation is obvious when you realize that r_i represents the proportion of the i^{th} variable to the total contribution of the block to the right hand side. Since each row in the tableau represents an equality which must be satisfied, and the components of the rows

don't change as the variables are adjusted, any changes in the variables must take place so that the right hand side equals the tableau times these variables. Therefore, the amount that the right hand side will be increased due to a unit increase in the block is given by the negative of the equation above, and the basic variables corresponding to each row must decrease by an equal amount to maintain feasibility.

The number of unit increases, Δ, which can be made in the block before driving a basic variable to zero must now be determined as:

$$\Delta = \min_{j=1,\ldots,N+1} \left(y_{b_j} / \gamma_{b_j} \middle| \gamma_{b_j} < 0 \right)$$

The ratio being minimized can easily be seen to equal the number of unit increases which can be made in the block before driving each basic variable to zero. The amount that any basic variable can decrease can only be as large as the value of the variable itself. By dividing each variable by the decrease in that variable for each unit increase in the block, the result is the number of unit increases which can be made before the variable is reduced by itself. Minimizing this value assures that feasibility will hold for all basic variables.

This need only be done if γ_{b_j} is negative, because if it is positive the basic variable will increase with increases in the block. If this is the case these variables don't constrain the increase of the block. If none of these values are negative, Δ is set to 1 and θ=100. Otherwise, set θ=1. Then determine the direction vector as follows:

$$d_i = r_i\Delta \quad , \quad \text{for i within the block being increased}$$

$$d_{b_j} = \gamma_{b_j}\Delta, \quad \text{for } j=1,\ldots,N+1$$

$$d_i = 0 \quad , \quad \text{elsewhere.}$$

then proceed to step 6.

e) A block of variables is being decreased towards zero. The ratios $r_i=y_i/\Lambda_k$, for all i within the block where k represents the block being decreased, are calculated for use in case the block must be increased in a future iteration. In this case the maximum feasible decrease in each variable in the block is merely the current value of each variable. Then each basic variable must increase by the sum of the corresponding row components times the variable values within the block. These values have already been checked in step three to insure that no basic variables are driven

negative. The direction vector is therefore calculated as follows:

$$d_{b_j} = \sum_i t_{ji} y_i \quad , j=1,\ldots,N+1$$

$$d_i = -y_i \quad , i \text{ within the block}$$

$$d_i = 0 \quad , \text{elsewhere.}$$

set $\theta = 1$ and go to step 6.

Step 6:

This step is used to find the optimal move along the direction vector. The direction vector defines the maximum possible feasible changes in the dual variables. However, a move of the maximum feasible amount isn't necessarily optimal. This is because the derivative of the objective function with respect to the direction vector changes as the dual variables change. In the regular Simplex this derivative reduces to the reduced costs which are constants over the move. Therefore, the maximum feasible move will also be optimal and all moves will be along an edge of the feasible region. For the Convex Simplex this is not the case and it is necessary to find the optimal move which may be within the region.

The location along the direction vector at any

point in the maximization is given by:

$$z_i = y_i + \tau d_i \quad , 0 < \tau \leq \theta$$

In order to maximize the objective funtion, the point where the derivative with respect to τ approaches zero must be found. This derivative for any value of is approximated by:

$$\sum_i G_i(z_i) \cdot d_i$$

where $G_i(z_i)$ is the gradient vector component given z_i. Since this is a continuous function in τ, a search along the range of τ can be used for this optimization.

Once the optimal value of τ has been determined, the new values of the dual variables are given by z_i. Sort this new dual vector in descending order and pivot the N+1 largest variables into the basis. If the value of τ is less than some preselected tolerance, proceed to step 7 to determine if the solution is consistant for the primal problem, otherwise return to step 1.

Step 7:

In this step the primal variables are estimated from the dual solution. This means solving the following set of linear equations:

$$\sum_j a_{ij} Z_j = -G_i \quad , \text{ for all i with } y_i > 0$$

where a_{ij} is the exponent of the j^{th} variable in the i^{th} primal term,.

and $Z_j = \ln(x_j)$, x_j is the j^{th} primal variable.

These equations can be seen to be the same as those used to calculate the primal variables using the regular geometric programming algorithm. In the regular GP the terms in the objective function are equated to the corresponding dual variables multiplied by the dual objective function value. The other terms are equated to the associated dual variables divided by the sum of the dual variables corresponding to the constraint which the term is in. The equations given for the Convex Simplex are the same as taking the natural logarithms of both sides of the equations just described. The left hand side of these equations are obviously equal, and a look at the equations used to describe G_i will easily show these to be equal to the negative of the log of the right hand side.

Now that these relationships have been established, all that is necessary to obtain the primal variables is to solve this system of linear equations and calculate x_j by taking the exponent of Z_j. Usually this system has more equations than unknowns, so either a least squares solution could be used, or equations could be eliminated

until a full rank set of equations is left. However, it's possible that a system of less than full rank may exist. In this case it is possible to increase the rank of the system by solving a subsidiary maximum problem. "A subsidiary maximum problem has essentially the same structure as the dual program, except that its objective function is modified by replacing the nonlinear terms corresponding to the positive components in the optimal vector by linear terms." (4) Problems which require subsidiary maximum solutions have been observed to be very rare (4) and will therefore not be treated in this thesis. For further details on this technique refer to the reference (1).

Now the values of x_j which have been calculated can be used to determine if the problem is optimal and feasible. These values are substituted into the primal objective function and constraints for this purpose. If the constraints are close enough to being feasible (less than one plus some tolerance) and the primal objective function value differs from the dual by less than some tolerance, the problem is solved. Otherwise, proceed to step 1.

This completes the algorithm.

3: An Example- Using the Modified Convex Simplex Method

The following problem, taken from Beightler and Phillips (2), will be solved in order to illustrate the use of this algorithm.

$$\text{Minimize } x_1^{-1}x_2^{-1}x_3^{-1}$$

subject to:

$$2x_1 + x_2 + 3x_3 \leq 1$$
$$x_1 + x_2 + x_3 \leq 1$$
$$x_1 + 3x_2 + 2x_3 \leq 1$$

Step 0: Initialization-

Initial Tableau

Basic Var	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	R_1	R_2	R_3	R_4	b
R_1	-1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
R_2	-1	1	0	0	1	0	0	1	0	0	0	1	0	0	0
R_3	-1	0	1	0	0	1	0	0	1	0	0	0	1	0	0
R_4	-1	0	0	1	0	0	1	0	0	1	0	0	0	1	0

The phase I Simplex is solved by using the above tableau to solve a problem with the following objective function:

$$\text{Minimize } R_1 + R_2 + R_3 + R_4$$

The final tableau after all artificial variables have been eliminated is:

Basic Var	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	b
u_1	1	0	0	0	0	0	0	0	0	0	1
u_2	0	1	0	0	1	0	0	1	0	0	1
u_3	0	0	1	0	0	1	0	0	1	0	1
u_4	0	0	0	1	0	0	1	0	0	1	1

The initial dual vector is therefore, (1,1,1,1, 0,0,0,0,0,0). A positive vector is computed by setting all nonbasic variables equal to one half of the minimum of the right hand side divided by the sum of the nonbasic columns for each row, which equals $\frac{1}{4}$. The basic variables are calculated by subtracting the sum of the nonbasic variables times the nonbasic columns from the right hand side for each row. The resulting initial positive dual feasible vector is (1,.5,.5,.5,.25,.25,.25,.25,.25, .25).

First Iteration-

First the nonbasic variable(s) to change must be determined. The initial objective function value is 103.5133 with an initial gradient of (-4.6397,1.7918,1.0986, 2.1972,1.0986,1.0986,1.0986,1.0986,2.1972,1.7918). The relative cost components are computed as follows:

$$c_1^* = -4.6397 - ((-4.6397)(1) + (1.7918)(0) + 0 + 0)$$
$$\vdots$$

$$c_5^* = 1.0986 - ((-4.6397)(0) + (1.7918)(1) + (1.0986)(0) + (2.1972)(0))$$

$$\vdots$$

$$c_{10}^* = 1.7918 - (0 + 0 + 0 + (2.1972)(1))$$

giving the following vector (0,0,0,0,-0.69315,0,-1.0986, -0.69315,1.0986,-0.4547).

The maximum relative cost is 1.0986 for u_9 and the minimum of the product of relative cost and dual variable is -0.2747 for u_7. Since 1.0986 is greater than 0.2747, variable u_9 is chosen for increase.

Now the direction vector is calculated.

$$d_9 = \min (- , - , 0.5 , -) = 0.5$$

The - indicates that $t_{js}=0$ for that basic variable. Because u_3 is the only basic variable with a nonzero element in the ninth column, $d_3=-0.5$ and all other basic elements in the direction vector equal zero. This gives a direction vector of (0,0,-0.5,0,0,0,0,0,0.5,0).

A search reveals that the optimal move along this direction vector is at 0.5811. The new dual vector is therefore; (1,0.5,0.20943,0.5,0.25,0.25,0.25,0.25,0.54057, 0.25), giving an objective function value of 122.069. Since the new value of u_9 is greater than u_3, u_9 is pivoted into the basis in the third row. This will not change the tableau given at the end of step 0 because

the ninth column is already in the proper form, but this will not be the case for many problems.

Second Iteration:

The new relative cost vector is (0,0,0.40746 X 10^{-8}, 0,-0.47783,-0.6549,-0.8833,-0.15038,0,0.1373). The C_i^* associated with the second constraint; C_5^*, C_6^*, C_7^*, are seen to all be negative, therefore it is possible that the corresponding constraint is inactive and a block decrease is indicated.

The feasibility of the block decrease is determined by calculating the direction vector components for the basic variables and determining if any of these basic variables could be driven negative. These basic components all turn out to be positive so that the block decrease is feasible with a direction vector of (0,.25,0,.25,-.25,-.25,-.25,0,.25,0). The maximum objective function is found with the block set all the way to zero with a value of 194.268.

Primal Variable Estimation:

After 41 iterations the following dual vector is found, (1.0,0.70235,0.28238,0.63902,0,0,0,0.29765,0.71762, 0.36098). Which gives a gradient of (-5.3121,1.531, 1.75,2.031,1.0986,1.0986,1.0986,1.531,1.75,2.031), and an objective function equal to 202.777.

Using G_2, G_3, and G_4, the following three independent equations result:

$$\ln x_1 = -1.53096$$

$$\ln x_2 = -1.75$$

$$\ln x_3 = -2.0311$$

giving:

$$x_1 = 0.21633$$

$$x_2 = 0.17377$$

$$x_3 = 0.13119$$

The primal objective function is 202.777 which agrees with the dual and the constraints have the following values respectively, 1.0, 0.52129, 1.000026, therefore the problem is solved.

4: Programming the "odified Convex Simplex "ethod-

The next step in the analysis is to write a computer program to solve geometric programs using the modified convex simplex (4). (The use of this program is detailed in appendix B) "ost of this process involves translating the steps in the reference (4) into computer code. All that need be addressed when doing this is some bookeeping which need not be discussed here. However, certain steps in the program require further explanation.

4.1: The I"SL (5) Library-

Two subroutines from the I"SL library (5) are used to help with some repetitive steps. "henever matrix multiplication is required, the subroutine V"ULFF is used. This subroutine multiplies two matrices in full storage mode giving a third full storage matrix as the result. The subroutine LEQT2F is used whenever it is necessary to solve systems of linear equations. This is needed when solving zero degree of difficulty problems, and for calculating the primal variables from the dual.

4.2: Objective Function Optimization-

At each iteration it is necessary to determine the optimal proportion of the direction vector for the move. This is done by finding the point where the derivative

of the objective function with respect to the direction vector approaches zero. The function FMIN from Forsythe, Malcolm, and Moler (6) is used to find this point. This function finds the minimum of a unimodal function between two points. In order to do this a function must be defined describing the derivative at any given point along the direction vector. This is done in function F in the program. This merely defines the derivative using the equations described in the given description of the algorithm.

4.3: Initialization-

A major modification of the convex simplex method involves the initial basic dual feasible solution as described earlier. A routine must be provided in this program to determine an initial solution which has the required properties. This is done by first using phase I of the two phase simplex method, as discussed earlier. If it is possible to solve this problem so that all artificial variables are nonbasic, the problem is considered canonical. If there is no feasible solution to the constraint set, at least one of the artificial variables will remain in the basis at optimality. In this case the appropriate message is printed and the program is terminated.

Once the initial basis is defined it is necessary to find a feasible positive vector for all dual variables. This is done by first summing the values of each row in the tableau except for the basic columns. Then the minimum of the current right hand side divided by one plus the row sum is determined. One half of this value is used as the starting value for all nonbasic dual variables. The values of the basic variables are then calculated by subtracting the product of the row sum times the nonbasic variables from the current right hand side. This procedure insures that all dual variables will be positive and that all basic variables will be larger than the nonbasic variables.

4.4: Stopping Conditions-

The final decision to be made concerns the stopping conditions for the algorithm. The program is checked for optimality and feasibility if the maximum relative cost and the minimum of the product of relative cost times the dual variables equals zero or if the step size along the direction vector at any iteration is less than some predetermined value. This step size was chosen to be 0.0001. To test for optimality the primal variables are calculated and the values of the primal objective function and constraints are determined. If

the dual objective function differs from the primal by less than 0.1% the problem is considered optimal and the solution is checked for feasibilty. If all of the primal constraints are less than 1.00001 the problem is considered feasible and the final solution is printed. When calculating the primal variables from the dual, only those equations corresponding to terms in active primal constraints are meaningful. An inactive primal constraint would have all dual variables equal to zero. Any dual variable less than 10^{-6} is considered to be zero for this transformation.

4.5: Zero Degrees of Difficulty-

As mentioned earlier, the subroutine LEQT2F (5) is used whenever a zero degree of difficulty problem is encountered. A completely separate section of the program is used to handle these problems. The linear equations solved are the initial dual constraints, which form a full rank set of equations if there are zero degrees of difficulty. Then the primal variables are calculated using the same routine as used in the main program.

5: Conclusions-

5.1: Advantages of the Modified Convex Simplex Method-

The modified convex simplex algorithm (4) provides a technique whereby high degree of difficulty posynomial geometric programs can be solved. One advantage of using this method is that it works out of the familiar Simplex tableau. In fact, the linear simplex method can be shown to be a special case of the convex simplex. Many programs already exist for solving the regular simplex method and the modified convex simplex only changes these in the determination of the variables to change. This makes it easier to program this algorithm and easier for operations researchers who already have knowledge of the simplex method to understand it.

This method will work on any problem which can be put into canonical form. Certain other solution methods require that inactive primal constraints be identified and eliminated from the problem or slack variables added before it can be solved. The convex simplex method makes no such requirements, although if inactive constraints are eliminated the method will converge more rapidly (2). However, the block decreases in the algorithm are equivalent to eliminating inactive primal constraints and therefore cause more rapid

convergence of the problem.

5.2: Limitations of the Method-

There are some limitations on this algorithm. The first being that it can only be used to solve posynomial problems. This just means that any problem must be in the form of a posynomial program before the algorithm is invoked. Techniques have been proposed which could transform signomial problems into posynomials (7). If the user were to make this transformation the resulting posynomial program could solved using this algorithm.

Another limitation is on degenerate problems, a degenerate problem being one in which a primal term can approach zero without causing any other term to approach infinity (2). This could lead to basic variables which are equal to zero and therefore non-convergence due to cycling. The result of this degeneracy on the dual is that it won't be possible to find an initial positive dual feasible vector.

This algorithm will only work on canonical (non-degenerate) problems. It may be desirable to try to identify degenerate problems initially so as not to spend time trying to solve an insoluble problem. The only sure way to identify degenerate problems is if a primal variable has all exponents of the same sign.

Otherwise there are no hard and fast rules to define a degenerate problem. In general a program with a much greater number of negative exponents is more likely to be degenerate than one with a balanced number of signs or with more positive than negative exponents (2). However, this is just a relative measure and the only way to be sure that a problem is not canonical is to attempt to solve it.

5.3: Computational Results-

Test problems were solved using this algorithm with up to 16 degrees of difficulty. (These problems are presented in appendix C) The results are summarized as follows:

Problem Number	DOD	Primal Variables	System Seconds	Number of Iterations
1	0	4	3.5	-
2	1	3	4.0	4
3	6	3	4.9	41
4A	10	7	6.8	45
4B	10	7	14.9	154
4C	10	7	20.0	254
5	16	4	7.7	68

In general, it is observed that solution time increases as degrees of difficulty and primal variables

increase. However, times may vary greatly even for problems with the same number of variables and degrees of difficulty. Problems 4A, B, and C were taken from a paper of geometric programming test problems (9). They differ only in the value of one exponent in the objective function, yet there are dramatic differences in solution time and number of iterations. This illustrates some of the difficulty in predicting the time needed to solve a problem just by observing properties of the problem.

Some general statements about the algorithm were made by Beck and Ecker. They found that "when the coefficients in the primal problem differ by many orders of magnitude, an extremely precise dual solution is necessary," (4) and therefore a longer solution time is required. Of course "many orders of magnitude" and "extremely precise" are subjective descriptions which don't lead to precise classification of problems. However, everything else being equal, a problem with coefficients which differ to a larger degree than another problem will probably take more time to solve.

"A final observation is that the more nonlinear the primal program, the easier it is to solve the dual program." (2) Nonlinear is also a subjective term, but

it basically refers to the number of exponents which are not equal to one or zero for each primal variable in each term.

In summary, although there are some basic rules which can be applied to try to predict the solution time for a given problem, there is a great variation among problems conforming to these rules. The only sure way to know how difficult a problem is to solve is to go ahead and attempt to solve it.

6: Areas of Future Study-

6.1: Generalized Geometric Programming-

A method has been proposed by Avriel, Dembo, and Passy (7) to transform signomial problems into posynomials, although the transformed problems still have the problem of local optimum. This method should be studied as to the possibility of using it to generalize the modified convex simplex method to allow solution of signomial problems.

The authors also present a technique to solve generalized geometric programs via a cutting plane algorithm. Some computational results were included which suggest that this technique could be used to solve very large (high degree of difficulty) problems with a relatively short computer times. These results indicate that further examination of this algorithm would be warrented.

6.2: Augmented Geometric Programming:

Another algorithm worth further study was proposed by McNamara (8). This procedure involves formation of an augmented problem with zero degrees of difficulty. When the augmented program is formed slack variables with unknown exponents are added to the problem. The algorithm involves solving a series of zero degree of

difficulty problems in order to estimate the optimal exponents. Computational experience with this technique suggests that it may be efficient for solving high degree of difficulty problems.

6.3: Surrogated Geometric Programming-

One final algorithm worth mentioning is presented in Beightler and Phillips (2). Specifically it is the Surrogated Geometric Programming algorithm. This algorithm also involves solving a series of zero degree of difficulty problrms. In this case the primal constraint set is replaced by a surrogate constraint which is a linear combination of the constraints. In this constraint surrogate multipliers which must sum to one are introduced and the algorithm consists of finding a feasible set of these multipliers. Basically a series of zero degree of difficulty problems are solved in order to move from superoptimality to feasibility by varying the surrogate multipliers. The only problem with this technique is that the speed by which it converges depends upon the initial choice for the multipliers and there is no set method of choosing them.

6.4: Other Areas of Study-

There are many other algorithms which can be used to solve posynomial geometric programs, however most

of these are variations of those already mentioned. A comparative study of these algorithms would be valuable in order to find one standardized solution method. With the realization of the value of geometric programming, the value of finding the most efficient technique for solving these problems would be significant.

Further study into the best method for solving signomial geometric programs would also be valuable. For years people have been faced with the problem of solving generalized nonlinear optimization problems. Many of these problems are in the geometric programming problem form or could be transformed easily into this form. Even functions which don't seem readily transformed could be approximated by a power series expansion with the accuracy desired determining how many terms to include. A generalized geometric programming algorithm would be a powerful technique for solution of these problems. Further study into developing such a technique could make it possible to solve some problems which previously had been almost impossible to solve.

REFERENCES

(1) Duffin, R.J., Peterson, E.L., and Zener, C., Geometric Programming - Theory and Applications, John Wiley and Sons, New York, N.Y., 1967.

(2) Beightler, C.and Phillips, D.T., Applied Geometric Programming, John Wiley and Sons, New York, N.Y., 1976.

(3) Zangwill, W.I., Nonlinear Programming: A Unified Approach, Prentice Hall, Englewood Cliffs, N.J., 1969.

(4) Beck, P.A. and Ecker, J.G., "A Modified Concave Simplex Algorithm for Geometric Programming," J. of Opt. Theory and Appl., Vol 15, No. 2, 1975.

(5) International Mathematical and Statistics Libraries, Inc., IMSL Library 3, Edition 6, 1977.

(6) Forsythe, G.E., Malcolm, M.A., and Moler, C.B., Computer Methods for Mathematical Computations, Prentice Hall, Englewood Cliffs, N.J., 1977.

(7) Avriel, M., Dembo, R., and Passy, U., "Solution of Generalized Geometric Programs," International Journal for Numerical Methods in Engineering, Vol 9, 1975.

(8) McNamara, J.R., "A Solution Procedure for Geometric Programming," Operations Research, Vol. 24, No. 1, 1976.

(9) Dembo, R.S., "A Set of Geometric Programming Test Problems and Their Solutions," Mathematical Programming, Vol. 10, 1976.

Appendix A

Theoretical Basis for Geometric Programming (1)

Geometric Programming is based on the relationship between the geometric and arithmetic means. The arithmetic mean of n numbers (or functions) U_i being:

$$A = \sum_{i=1}^{n} (U_i/n)$$

and the geometric mean is defined by:

$$G = \prod_{i=1}^{n} U_i^{1/n}$$

The relationship between these values is known as the geometric inequality and is represented as:

$$A \geq G \qquad (1)$$

as long as U_i is any non-negative term.

This inequality is easily shown to be true by the following operation on an obviously true relationship (for n=2):

$$(U_1 - U_2)^2 \geq 0$$

$$U_1^2 - 2U_1U_2 + U_2^2 \geq 0$$

$$U_1^2 + 2U_1U_2 + U_2^2 \geq 4U_1U_2$$

$$\tfrac{1}{4}U_1^2 + \tfrac{1}{2}U_1U_2 + \tfrac{1}{4}U_2^2 \geq U_1U_2$$

The square root of this last inequality gives inequality (1), for n=2. For n=4 the inequality becomes:

$$\tfrac{1}{4}U_1 + \tfrac{1}{4}U_2 + \tfrac{1}{4}U_3 + \tfrac{1}{4}U_4 \geq U_1^{\frac{1}{4}}U_2^{\frac{1}{4}}U_3^{\frac{1}{4}}U_4^{\frac{1}{4}}$$

If $U_2 = U_3 = U_4$ this can be rewritten as:

$$\tfrac{1}{4}U_1 + (3/4)U_2 \geq U_1^{\frac{1}{4}}U_2^{(3/4)}$$

Therefore the relationship holds even for weighted means as long as the weights sum to one.

The Dual-

If a generalized posynomial with terms:

$$U_i = c_i t_1^{a_{i1}} t_2^{a_{i2}} \cdots t_m^{a_{im}}$$

is to be minimized, the geometric inequality can be stated as:

$$d_1U_1 + d_2U_2 + \cdots + d_nU_n \geq U_1^{d_1} U_2^{d_2} \cdots U_n^{d_n}$$

where d_i are arbitrary positive weights which sum to one. If we let $u_i = U_i d_i$, the inequality becomes:

$$u_1 + u_2 + \cdots + u_n \geq (u_1/d_1)^{d_1}(u_2/d_2)^{d_2}\cdots(u_n/d_n)^{d_n}$$

This right hand side is termed the pre-dual function V. If the original terms with variable t_j and coefficients c_i are substituted into V it becomes:

$$V(d,t)=(c_1/d_1)^{d_1}(c_2/d_2)^{d_2}\cdots(c_n/d_n)^{d_n} t_1^{D_1}\cdots t_m^{D_m} \qquad (2)$$

where the exponents D_j are:

$$D_j = \sum_{i=1}^{n} d_i a_{ij} \quad , \quad j=1,\ldots,m$$

where a_{ij} is the exponent of t_j in the i^{th} term.

If the weights, d_i, are chosen so that all D_j equal zero, V(d,t) no longer depends upon t_j and becomes the Dual function v(d):

$$v(d) = (c_1/d_1)^{d_1}(c_2/d_2)^{d_2}\ldots(c_n/d_n)^{d_n} \qquad (3)$$

It can be shown that the values of d_i which make the D_j vanish also give the upper bound on v(d). The form of the geometric inequality makes it obvious that if an upper bound on v is found, it must also be a lower bound on the left hand side. Therefore to minimize the original function (the left hand side) it is only necessary to find the point where the D_j equal zero, which is defined by the following equations:

$$\sum_{i=1}^{n} d_i = 1 \qquad (4)$$

$$D_j = 0 \quad , \quad j=1,\ldots,m \qquad (5)$$

Equation (4) above normalizes the weights to 1 and is known as the normality condition. The equations represented by (5) are termed the orthogonality condition.

The use of these conditions to solve posynomial problems can best be illustrated using an example.

Suppose we have the following problem:

$$\text{Minimize} \quad \frac{40}{t_1 t_2 t_3} + 40 t_2 t_3$$

subject to:

$$\frac{t_1 t_3}{2} + \frac{t_1 t_2}{4} \leq 1$$

Substituting u_i for each term in the functions this problem becomes:

$$\text{Min} \quad g_0 = u_1 + u_2$$

$$\text{S/T} \quad g_1 = u_3 + u_4 \leq 1$$

Because there is a constraint, a more general form must be used where all of the weights are no longer normalized. If we use w_i for the unnormalized weights and k as their sum, the relationship between the normalized and unnormalized weights is:

$$w_i = kd_i \quad , \quad i=1,2,3,4$$

Substituting w_i/k for d_i in the geometric inequality gives:

$$u_1 + u_2 + \ldots + u_4 \geq (u_1/w_1)^{w_1/k} \ldots (u_n/w_n)^{w_n/k} \; k$$

Taking both sides to the k^{th} power gives:

$$(u_1 + \ldots + u_4)^k \geq (u_1/w_1)^{w_1} \ldots (u_n/w_n)^{w_n} \; k^k$$

Using this inequality on the problem in question we get:

$$g_0^{k_0} \geq (u_1/w_1)^{w_1} (u_2/w_2)^{w_2} \; k_0^{k_0}$$

$$1 \geq g_1^{k_1} \geq (u_3/w_3)^{w_3}(u_4/w_4)^{w_4}\, k_1^{k_1}$$

Multiplying the first inequality by both sides of the second gives:

$$g_o^{k_o} \geq (u_1/w_1)^{w_1}(u_2/w_2)^{w_2}(u_3/w_3)^{w_3}(u_4/w_4)^{w_4}\, k_o^{k_o}\, k_1^{k_1}$$

If we still normalize k_o to 1 the predual function is:

$$g_o(t) \geq (u_1/w_1)^{w_1} \ldots (u_4/w_4)^{w_4}\, k_1^{k_1}$$

The dual function from this predual is:

$$(c_1/w_1)^{w_1}(c_2/w_2)^{w_2}(c_3/w_3)^{w_3}(c_4/w_4)^{w_4}\, k_1^{k_1}$$

Since $k_1^{k_1} = k_1^{w_3+w_4} = k_1^{w_3} k_1^{w_4}$, this becomes:

$$(c_1/w_1)^{w_1}(c_2/w_2)^{w_2}(c_3 k_1/w_3)^{w_3}(c_4 k_1/w_4)^{w_4}$$

with the normality and orthogonality conditions of:

$$w_1 + w_2 = 1$$

$$-w_1 + w_3 + w_4 = 0$$

$$-w_1 + w_2 + w_4 = 0$$

$$-w_1 + w_2 + w_3 = 0$$

which gives the solution:

$w_1 = 2/3$ $\quad w_2 = 1/3$ $\quad w_3 = 1/3$ $\quad w_4 = 1/3$.

Therefore the final solution can be had from the geometric inequality as:

$$g_0(t) \geq \left(\frac{40}{2/3}\right)^{2/3}\left(\frac{40}{1/3}\right)^{1/3}\left(\frac{2/3}{2/3}\right)^{1/3}\left(\frac{2/3}{4/3}\right)^{1/3} = 60$$

Therefore the minimum possible feasible value for g_0 is 60 and the rpoblem has been solved.

If the values of t_i are desired, it is necessary to go back to the meaning of the optimal weights. Each weight represents the contribution of the corresponding term to the equation which it is in. Therefore, for the problem just solved the following equalities define the primal variables:

$$\frac{40}{t_1 t_2 t_3} = (2/3)(60) = w_1 g_0$$

$$40 t_2 t_3 = (1/3)(60) = w_2 g_0$$

$$\frac{t_1 t_3}{2} = (1/3)/(2/3) = w_3/k_1$$

$$\frac{t_1 t_2}{4} = (1/3)/(2/3) = w_4/k_1$$

which can be solved to give:

$t_1 = 2$ $\quad t_2 = 1$ $\quad t_3 = 0.5$.

(The information in this appendix comes from Duffin, Peterson, and Zener (1). For more details, please refer to this reference)

Appendix B

Using the Program

Before the program can be used the problem must be in the proper form. This program requires that the problem first must be in canonical form. This means that the objective function must be a minimization and all constraints must be less than or equal to one. No strict equality constraints are allowed nor can any coefficient in the problem be negative. The program has been dimensioned so that no more than 19 primal variables are allowed. Also, only 19 constraints and 20 terms per constraint can be used. Finally no more than 59 total terms can be in the primal problem.

The program has been designed to run in the batch mode. The user must provide four types of data cards which describe the problem to be solved. The program will then either conclude that the problem is not canonical (degenerate) or it will find the optimal solution and print the values of the optimal dual and primal variables and objective functions.

The data cards needed are as follows:

I) Problem Description-

```
      0          1
      5          5
   NVAR      NCNSTR

   FORMAT (I5,5X,I5)
```

where: NVAR = number of primal variables (integer)

NCNSTR = number of primal constraints (integer)

II) Number of terms-

FORMAT (2014)

On this card the number of terms in the primal objective function and constraints must be listed in order in the format described above.

III) Coefficients-

FORMAT (5E16.10)

Each card has the coefficients of one primal constraint or the objective function. These cards all adhere to the above format. If there are more than five terms for any given equation just use as many cards as necessary until all the coefficients are listed. For example, if one constraint had twelve terms, then three cards would be needed with only two values on the third card.

IV) Exponents-

These cards follow the same format as for the coefficients, however each card will have the exponents of all variables for a given term. An exponent must be included for every primal variable for every term. If a primal variable is not present in a given term, an exponent of zero must be placed in the proper position.

Once all the data cards are typed they must be put in the proper order. First must be the problem description card followed by the number of terms card. Next the coefficient card for the objective function is input followed by the exponent cards for all terms in the objective function. Finally, the coefficient cards for the primal constraints are input followed by the corresponding exponent cards.

An example of how to use the program follows.

The data for the following problem is given on the following page:

Minimize $x_1^{-1}\ x_2^{-1}\ x_3^{-1}$

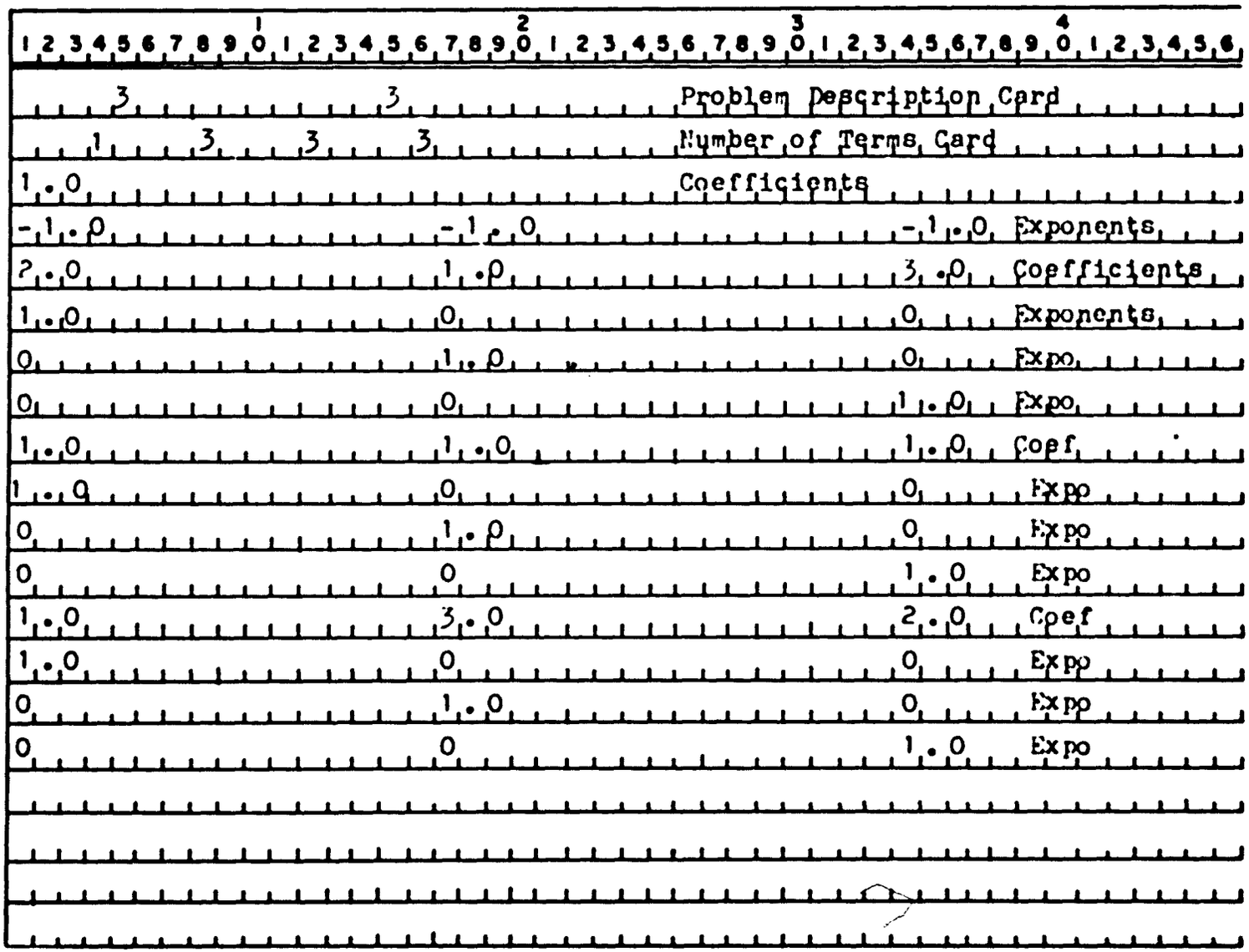
Subject to:

$$2x_1 + x_2 + 3x_3 \leq 1.0$$

$$x_1 + x_2 + x_3 \leq 1.0$$

$$x_1 + 3x_2 + 2x_3 \leq 1.0$$

$$x_i \geq 0.0 \quad , \text{ for } i=1,2,3$$


1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1 2 3 4 5 6 7 8 9 40 1 2 3 4 5 6
1 3 3 Problem Description Card
2 1 3 3 3 Number of Terms Card
3 1.0 Coefficients
4 -1.0 -1.0 -1.0 Exponents
5 2.0 1.0 3.0 Coefficients
6 1.0 0 0 Exponents
7 0 1.0 0 Expo
8 0 0 1.0 Expo
9 1.0 1.0 1.0 Coef
10 1.0 0 0 Expo
11 0 1.0 0 Expo
12 0 0 1.0 Expo
13 1.0 3.0 2.0 Coef
14 1.0 0 0 Expo
15 0 1.0 0 Expo
16 0 0 1.0 Expo
17
18
19
20

The output from the program for this problem would be:

EQUATION	TERM	DUAL VARIABLE VALUE
0	1	1.00000
1	1	.70?35
1	?	.?8?38
1	3	.63902
?	1	.00000
?	?	.00000
?	3	.00000
3	1	.?9765
3	?	.71?62
3	3	.36098

DUAL OBJECTIVE FUNCTION VALUE = ?02.77702

PRIMAL VARIABLES

NUMBER	VALUE
1	.21627
?	.17381
3	.13119

PRIMAL OBJECTIVE FUNCTION VALUE = 202.77702

Appendix C

Test Problems

Problem Number One (2)-

Minimize $x_3^{0.8} x_4^{1.4}$.

Subject to:

$$x_1^{-2} x_3^{-1} + x_2 x_3^{-1} \leq 1$$

$$x_1 x_4^{-1} + x_2^{-1} x_4^{-1} \leq 1$$

Solution:

$x_1 = 0.05556$ $\qquad$ $x_2 = 108.00$

$x_3 = 432.00$ $\qquad$ $x_4 = 0.06481$

Objective function = 2.7844

Problem Number Two (2)-

Minimize $0.3x_1 x_3^{1.2} + 0.5x_1^{-1.5} x_2^{-1.2} x_3^{-1.4} + 0.2x_2^{1.3}$

Subject to:

$$0.8x_1 x_3 \leq 1$$

$$1.2x_1 x_2^{-1} \leq 1$$

Solution:

$x_1 = 1.223$ $\qquad$ $x_2 = 1.468$

$x_3 = 0.861$

Objective function = 0.9189

Problem Number Three (2)-

Minimize $x_1^{-1}x_2^{-1}x_3^{-1}$

Subject to:

$$2x_1 + x_2 + 3x_3 \leq 1$$

$$x_1 + x_2 + x_3 \leq 1$$

$$x_1 + 3x_2 + 2x_3 \leq 1$$

Solution:

$x_1 = 0.21633$ $\qquad$ $x_2 = 0.1738$

$x_3 = 0.13118$

Objective Function = 202.777

Problem Number Four A, B, C (9)-

Minimize $10.0x_1x_2^{-1}x_4^2x_6^{-3}x_7^a + 15.0x_1^{-1}x_2^{-2}x_3x_4x_5^{-1}x_7^{-\frac{1}{2}}$

$+ 20.0x_1^{-2}x_2x_4^{-1}x_5^{-2}x_6 + 25.0x_1^2x_2^2x_3^{-1}x_5^{\frac{1}{2}}x_6^{-2}x_7$

Subject to:

$0.5x_1^{\frac{1}{2}}x_3^{-1}x_6^{-2}x_7 + 0.7x_1^3x_2x_3^{-2}x_6x_7^{\frac{1}{2}} + 0.2x_2^{-1}x_3x_4^{-\frac{1}{2}}x_6^{2/3}x_7^{\frac{1}{2}} \leq 1$

$1.3x_1^{-\frac{1}{2}}x_2x_3^{-1}x_5^{-1}x_6 + 0.8x_3x_4^{-1}x_5^{-1}x_6^2 + 3.1x_1^{-1}x_2^{\frac{1}{2}}x_4^{-2}x_5^{-1}x_6^{1/3} \leq 1$

$2.0x_1x_3^{-3/2}x_5x_6^{-1}x_7^{1/3} + 0.1x_2x_3^{-\frac{1}{2}}x_5x_6^{-1}x_7^{-\frac{1}{2}}$

$+ 1.0x_1^{-1}x_2x_3^{\frac{1}{2}}x_5 + 0.65x_2^{-2}x_3x_5x_6^{-1}x_7 \leq 1$

$0.2x_1^{-2}x_2x_4^{-1}x_5^{\frac{1}{2}}x_7^{1/3} + 0.3x_1^{\frac{1}{2}}x_2^2x_3x_4^{1/3}x_5^{-2/3}x_7^{\frac{1}{2}}$

$+ 0.4x_1^{-3}x_2^{-2}x_3x_5x_7^{3/4} + 0.5x_3^{-2}x_4x_7^{\frac{1}{2}} \leq 1$

Problem A: a=-0.25 B: a=0.125 C: a=0.50

Solution:

Variable	A	B	C
Obj. Func.	1809.7615	911.87957	543.66638
x_1	2.8566276	3.8955214	4.3919085
x_2	0.61083257	0.8086847	0.8546317
x_3	2.1503944	2.6626285	2.8416293
x_4	4.7171337	4.2983005	3.4013674
x_5	1.0002048	0.85357785	0.7227534
x_6	1.3487370	1.0953123	0.87052969
x_7	0.03160686	0.02730898	0.02464651

Problem Number Five (2)-

$$\text{Minimize } 20x_1+10x_2+30x_3+15x_4 + 1500x_1^{-1}x_2^{-1}x_3^{-1}x_4^{-1}$$

Subject to :

$$10x_1 + 5x_2 + 5x_3 + x_4 \leq 60$$

$$2x_1 + 6x_2 + 8x_3 + 4x_4 \leq 37$$

$$4x_1 + 3x_2 + 7x_3 + 14x_4 \leq 45$$

$$x_1 + x_2 + x_3 + x_4 \leq 15$$

Solution:

$x_1 = 2.64$ $x_2 = 2.68$

$x_3 = 1.33$ $x_4 = 1.22$

Objective Function = 267.98

BIOGRAPHY

Lewis Craig Chasalow

Personal-

Born: July 11, 1956 - Washington, D.C.

Parents: Dr. Ivan and Mrs. Carol Chasalow

Single

Height: 6' 4" Weight: 195 lbs.

Education-

School	Date Entered	Date Graduated	Degree
Hanover Twp. Public S.	1961	1970	-
Whippany Park H.S.	1970	1974	Diploma
Lehigh University	1974	1978	B.S.I.E.
Lehigh University	1978	-	-

Honors-

At Lehigh University: Freshman Honors, Sophmore Honors, Dean's List, Graduation with Highest Honors.

Member: Phi Eta Sigma, Alpha Pi Mu, Tau Beta Pi.

Offered Presidential Fellowship from Virginia Polytechnic Institute and State University.

Accepted Gotshall Fellowship from Lehigh University to study for a Master of Science in Industrial Engineering.