Journal of Engineering Mathematics, Vol. 9, No. 1, January 1975
Noordhoff International Publishing–Leyden
Printed in The Netherlands

Discrete solutions to engineering design problems

J. J. DINKEL and G. A. KOCHENBERGER

Department of Management Science and Organizational Behavior, The Pennsylvania State University, University Park, Pennsylvania 16802, *U.S.A.*

(Received April 22, 1974)

SUMMARY

A method of obtaining discrete and/or integer valued solutions to non-linear design problems is presented. The general framework is that of geometric programming which is combined with the Branch and Bound Method. Recently developed computational procedures are described and are used to demonstrate the feasibility of the above method.

1. Introduction

This paper considers an extension of optimization techniques in the area of optimal design; in particular an extension of *geometric programming* [9] to deal with design problems with discrete valued variables. The motivation of this research lies in the fact that in many design situations some or all of the design variables are not continuous in nature; for example, some variables may be *integer valued* (number of sides of a container, number of tubes in a heat exchanger, etc.) and other variables may be *discrete valued* (thickness of steel plate, diameter of pipes, etc.). While geometric programming has been shown to be widely applicable in engineering design problems [1, 3, 5, 15, 16], it,as most optimization techniques, requires that the variables be continuous in nature. A general practice is to solve the design problem as if all variables were continuous and *round* the optimal solution to discrete or integer valued solution where necessary.

It became apparent in integer linear programming that the rounding operation has two serious drawbacks: (1) the rounded solution may not be feasible and (2) even if it is feasible it may be far from optimal [12]. The fact that we are dealing with non-linear functions will serve to exaggerate these difficulties. To obtain integer solutions in linear programming in an efficient manner the *Branch and Bound* [4, 12, 14] method has been successfully implemented. Since the Branch and Bound method is a recursive technique and requires the solution of a sequence of problems it has not received a lot of attention with respect to non-linear models; particularly in light of the potential effort required to solve each subproblem.

However, in light of recent developments for solving geometric programs, we describe an efficient procedure for obtaining discrete and/or integer solutions to non-linear optimization problems. This procedure is a combination of a sequential linear programming method for obtaining optimal solutions to geometric programs and the above mentioned Branch and Bound technique.

Section 2 gives brief descriptions of the LP approach to geometric programs and the Branch and Bound technique. Section 3 presents the combined algorithm and some examples. We conclude with a discussion of the extension of the results to the more general polynomial programs of Avriel and Williams [2, 3].

For the most part the paper is self contained. Most current texts dealing with engineering design contain explanatory material on GP [5, 15, 16]. The Branch and Bound method as described in the linear programming context applies here as well [12].

2. Basic concepts

The class of problems known as ordinary geometric programs [9] is characterized as:

$$\begin{aligned} &\min g_0(t), \\ &\text{s.t. } g_k(t) \leqq 1, \quad k=1, \ldots, p, \\ &t=(t_1, \ldots, t_m)>0, \end{aligned} \tag{1}$$

where

$$g_k(t)=\sum_{i=m_k}^{n_k} c_i \prod_{j=1}^{m} t_j^{a_{ij}}, \quad k=0, 1, \ldots, p, \tag{2}$$

are *posynomials* for $c_i>0$. The a_{ij} are arbitrary real numbers and the m_k, n_k are indices which consecutively number the terms of each function, i.e., $1=m_0, \ldots, n_0$ number the objective function terms; $n_0+1=m_1, \ldots, n_1$ for the first constraint and so on with $n_p=n$ for a total of n terms.

Beside the applicability of the above general model to many engineering design problems, another attractive feature of geometric programming is the relative ease of solution via the associated dual geometric program [9]. Several algorithms have been proposed to solve the dual program, which consists of maximizing a concave function over a set of linear constraints, and construct the primal solution from algebraic relationships. One disadvantage of a gradient based algorithm is the undefinediness of certain components of the gradient of the dual function in the presence of inactive (loose) primal constraints. For a discussion of these difficulties see, for example, [9].

We present the outline of an algorithm which avoids the above difficulties and reduces the solution of a geometric program to the solution of a sequence of linear programs. The following is self contained, however, the reader is referred to [6, 7, 8] for complete details. The algorithm is of the general form of a *cutting plane method* [13, 17]. Such methods deal with a linearized version of the problem and converge to the optimal solution from *outside the feasible region* by the addition of appropriate hyperplanes (cutting planes). As originally proposed the linearization is based on the 1st order Taylor Series expansion of the non-linear functions at the current operating point t^k. That is, a function $g(t)$ is approximated, at the point t^k, by:

$$g(t^k)+\nabla g(t^k)^T(t-t^k). \tag{3}$$

Extensions of the original development have been given to show the applicability of the method to other than convex functions [17].

The method to be described here is based on the *arithmetic-geometric inequality*, equation 23 [2, 8], and as a result avoids the gradient computations of (3). To illustrate the basic procedure consider the function:

$$g(t)=3t_1^{-1}t_2t_3^{-2}+t_2^{-1}t_3+2t_1t_2^{-\frac{1}{2}}=u_1(t)+u_2(t)+u_3(t). \tag{4}$$

The arithmetic-geometric inequality states that if $\bar{t}>0$ then

$$g(t)\geqq \Pi\,[u_i(t)/\varepsilon_i]^{\varepsilon_i}\equiv g(t;\varepsilon), \tag{5}$$

where

$$\varepsilon_i=u_i(\bar{t})/g(\bar{t}), \qquad i=1, 2, 3. \tag{6}$$

So if $\bar{t}_1=\bar{t}_2=\bar{t}_3=1$ then applying (5)–(6) we obtain the monomial form:

$$\left[\frac{3t_1^{-1}t_2t_3^{-2}}{3/6}\right]^{3/6}\left[\frac{t_2^{-1}t_3}{1/6}\right]^{1/6}\left[\frac{2t_1t_2^{-\frac{1}{2}}}{2/6}\right]^{2/6}=6^{1/2}6^{1/6}6^{2/6}t_1^{-1/6}t_2^{1/6}t_3^{-5/6}. \tag{7}$$

Taking the natural logarithm of (7) and making the change of variables $Z_j=\log t_j$ we obtain the *linear form*

$$\bar{g}(Z)=\log 6-\tfrac{1}{6}Z_1+\tfrac{1}{6}Z_2-\tfrac{5}{6}Z_3. \tag{8}$$

Thus we have linearized a posynomial form (4) by employing an algebraic inequality (5) and a log transformation. There are several points of interest concerning this approach:

1. The method is exterior; that is for constraints of the form $g_k(t) \leqq 1$ it follows from (5) that $1 \geqq g_k(t) \geqq g_k(t;\varepsilon)$. Thus the minimum value of the objective function is approached from below and the approximations, $g_k(t;\varepsilon)$, are at best *tangent* to the solution space.

2. The variables, Z_j, are unrestricted in sign, so in terms of solving such a system it is more efficient to deal with the dual system.

Applying the above notions to a posynomial GP of the form:

$$
\begin{aligned}
A\colon\ & \min t_0\,, \\
& \text{s.t. } t_0^{-1} g_0(t) \leqq 1\,, && (9)\\
& \qquad g_k(t) \leqq 1\,, \quad k = 1, \ldots, p\,, && (10)\\
& \qquad 0 < L_j \leqq t_j \leqq U_j < \infty\,, \quad j = 0, 1, \ldots, m\,,
\end{aligned}
$$

where the objective function, $g_0(t)$, has been treated as an additional constraint and lower and upper bounds have been introduced for each variable, we obtain the monomial program

$$
\begin{aligned}
A(\varepsilon)\colon\ & \min t_0\,, \\
& \text{s.t. } t_0^{-1} g_0(t;\varepsilon) \leqq 1\,, \\
& \qquad g_k(t;\varepsilon) \leqq 1\,, \quad k = 1, \ldots, p\,, \\
& \qquad 0 < L_j \leqq t_j \leqq U_j < \infty\,, \quad j = 0, 1, \ldots, m\,,
\end{aligned}
$$

where

$$
g_k(t;\varepsilon) = \prod_{i=m_k}^{n_k} (u_i(t)/\varepsilon_i)^{\varepsilon_i}\,,
$$

$$
\varepsilon_i = u_i(\bar{t})/g_k(\bar{t})\,, \quad m_k \leqq i \leqq n_k\,, \quad k = 0, 1, \ldots, p \text{ for } \bar{t} > 0\,.
$$

Applying the log transformation and the change of variables $Z_j = \log t_j$ we obtain the *linear program*

$$
\begin{aligned}
A_L(\varepsilon)\colon\ & \min Z_0\,, \\
& \text{s.t. } -Z_0 + C_0 + \sum_{i=1}^{n_0} \sum_{j=1}^{m} \varepsilon_i a_{ij} Z_j \leqq 0\,, \\
& \qquad C_k + \sum_{i=m_k}^{n_k} \sum_{j=1}^{m} \varepsilon_i a_{ij} Z_j \leqq 0\,, \quad k = 1, \ldots, p\,, \\
& \qquad\qquad Z_j \geqq \log L_j\,, \\
& \qquad\qquad Z_j \leqq \log U_j\,, \quad j = 0, 1, \ldots, m\,,
\end{aligned}
$$

where

$$
C_k = \log \prod_{i=m_k}^{n_k} (C_i/\varepsilon_i)^{\varepsilon_i}\,.
$$

In the algorithm that follows, the statement "Solve the LP", should be interpreted as solving the dual LP to $A_L(\varepsilon)$. As mentioned earlier this is more efficient since the Z_j are unrestricted in sign. In fact the method used by the authors also has generalized bounding capabilities which increases efficiency due to the presence of bounds on each variable.

The algorithm is:

STEP 0. For an initial point t^0 (feasible or infeasible), solve the linear program $A_L(\varepsilon^0)$.

STEP 1. If the optimal LP solution, t^*, is feasible with respect to (9)–(10), stop with the optimal solution to A being t^*.

(a) Otherwise, determine the most violated constraint; that is:

$\max\{k : g_k(t^*) > 1\}$

and go to STEP 2.

STEP 2. Suppose $k = 1$ is the result of (a); then form the cutting plane $g_1(t; t^*)$ and adjoin its linearized version to the original LP to yield $A_L(\varepsilon^1)$ and return to STEP 1.

The above approach provides a sequential LP method for obtaining solutions to geometric programs. Also, as will be discussed in the next section, this approach is particularly well suited to the Branch and Bound technique. As we will show the combination of these techniques provides an efficient procedure for obtaining integer and discrete solutions to non-linear design problems.

We present a brief discussion of the Branch and Bound technique; the reader is referred to [12, 14] for complete details. Branch and Bound, an enumerative scheme for obtaining optimal solutions in a systematic way, involves two basic operations:

1. *Branching*—an operation which divides the feasible solution space into a collection of subsets.
2. *Bounding*—the establishment of bounds on the objective function over the subsets of solutions determined by the branching operation. It is this operation which reduces the number of branches which are completely evaluated.

The version to be discussed here is to embed the discrete valued problem in a less restrictive solution space; the variables are assumed continuous, and then branch on the discrete and integer valued variables.

The branching operation divides the elements of the given feasible solution space into disjoint collections of points which include all the original feasible points. Specific branching rules can vary according to the problem at hand and in the next section we choose to branch on the discrete variables initially and then form secondary branches on integer variables.

The bounding operation provides upper and lower bounds on the value the optimal value of the objective function and on the value of the objective function over the subsets determined by the branching operation. These bounds play two important roles:

(1) The lower bound (optimal continuous solution) provides a convenient test concerning the feasibility of the various branches. For example if the upper bound over a branch is less than the continuous optimal solution that branch can be discarded as being infeasible.

(2) The upper and lower bounds on the potential subsets of discrete solutions provides a means of eliminating branches without completely evaluating the entire branch. Once an upper bound on the value of the objective function (the value of the best feasible discrete solution identified so far) is known, it is used to eliminate branches by comparing it with the lower bound of a branch. The lower bound can be the continuous solution and if the lower bound on a branch *exceeds* the current (upper) value of the objective function that branch is eliminated from consideration.

It is important to note that the above described procedure lends a sequential nature to Branch and Bound methods and as a result *reduces* the number of potential solutions that must be evaluated. Furthermore, Branch and Bound methods are optimal seeking; that is, they will locate the global optimal solution. In the next section we show that the branching operation is particularly suited to the previous cutting plane algorithm to obtain integer solutions to non-linear problems.

Before presenting the algorithm we give a small example to motivate the development. Consider the optimal design of a ventilating shaft [15] which is to be polygonal with sides of equal length and have a minimal cross sectional area of 600 sq. ft. The costs of construction involve an excavation cost ($\$35nL+\$3000n$) and the cost of concrete ($\$200nL+\$20nL^2$), where n is the number of sides (integer) and L is the length per side. The resulting design problem is:

$$\min\ 235nL+3000n+20nL^2\,,$$

$$\text{s.t.}\quad 1/4nL^2\cot(\Pi/n)\geqq 600 \quad \text{(cross sectional area)}\,, \tag{11}$$

$$n\geqq 3\,, \tag{12}$$

$$L>0\,.$$

To convert the above model to a posynomial GP we write (11) as

$$2400n^{-1}L^{-2}/\cot(\Pi/n)\leqq 1$$

TABLE 1

*Iteration**	*n*	*L*	*Cost*	*Execution time (sec)*†
0**	10	25	75000.0	
1	20	5.659	50000.0	
2	7.463	11.951	54935.9	
3	10.530	8.371	64288.5	
4	8.846	10.024	64481.7	
5	8.123	10.949	64576.5	
6	8.137	10.949	64686.9	
7	7.805	11.440	64782.9	
8	7.969	11.192	64818.8	
9	7.887	11.315	64821.8	
10	7.887	11.315	64824.9	
11	7.846	11.377	64827.5	
12	7.867	11.346	64827.5	
13	7.877	11.330	64827.5	
14	7.8818	11.3227 $n \leqq 7$	64827.5	0.2640
15	7.0000	12.847	57193.2	
16	7.0000	12.8467 $n \geqq 8$	65238.0	0.3540
17	20.0000	5.659	50000.0	
18	8.0000	11.1463	56769.1	
19	8.0000	11.1463	64833.2	
20	8.0000	11.1463	64833.2	0.4610

* Each iteration corresponds to the solution of an LP, the construction of a cutting plane and the formulation of the next LP in the sequence.
** Iteration 0 corresponds to the starting point.
† IBM 370/165.

and then replace $\cot(\Pi/n)$ by its Taylor Series expansion to obtain

$$2400n^{-1}L^{-2}\left[\frac{\Pi}{n}+\frac{1}{3}\left(\frac{\Pi}{n}\right)^3+\frac{2}{15}\left(\frac{\Pi}{n}\right)^5\right] \leqq 1\,. \tag{13}$$

Ignoring, for the moment, the integer restriction on n we obtain the global solution:

$$n^* = 7.8818\,,$$

$$L^* = 11.3227\,,$$

$$\cos t = 64827.5\,.$$

We now branch on the integer variable n as: $n \geqq 8$ and $n \leqq 7$ which yields the solutions:

$n^* = 8$	$n^* = 7$
$L^* = 11.1463$	$L^* = 12.8467$
$\cos t = 64833.2$	$\cos t = 65238.0$

As will be shown the algorithm requires only a modification of (12) to perform these branch operations. Also the continuous solution provides a convenient starting point; that is, since the algorithm does not require feasible starting points we need only modify (12) and continue iterating. Table 1 presents the computational results. We should also note that we are not making full use of the Branch and Bound method since there is only one variable (n) to brnach on.

3. Combined method and extension to polynomial programs

We now describe the combination of Branch and Bound methods with the linear programming

algorithm for obtaining discrete solutions to non-linear design problems. It should be noted that we do not develop specific branching rules since such rules would vary according to the problem at hand. It has been pointed out that the efficiency of Branch and Bound methods in the face of many discrete or integer variables depends heavily on an efficient development of branching rules.

The basis of the development here is to illustrate how the Branch and Bound method can be incorporated efficiently within an algorithm for solving non-linear problems. That is, the algorithm is not dependent on a specific set of branching rules. The approach is to embed the discrete or integer model in a continuous model and once the continuous solution has been obtained develop branches on the discrete variables and (if necessary) on the integer variables. Schematically, the method is indicated in Figure 1.

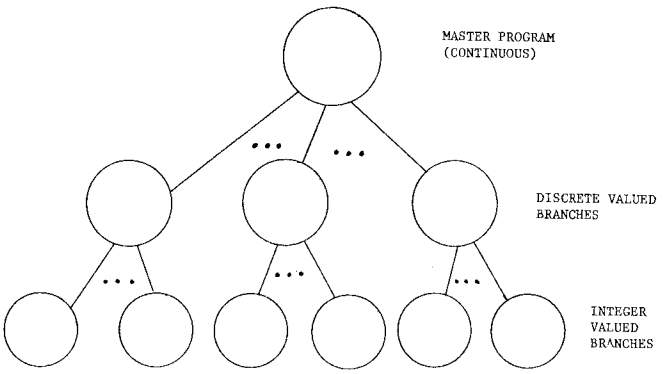

Figure 1.

In terms of combining the GP algorithm and the Branch and Bound method we assume the algorithm described in the previous section has been used to determine the optimal continuous solution, t^*. Since t^* is in fact the global optimum $g_0(t^*)$ provides a convenient test for feasibility of discrete solutions.

Assuming the optimal solution is not discrete valued we select those variables which are to be discrete valued and assign to them their respective values; thus creating discrete valued branches. If the number of such variables and values were small we might initially solve several programs with these values rather than the continuous problem. The values of the programs thus obtained provide lower bounds on the various branches and will be used in the elimination of other branches.

Assuming there are now integer valued variables to be determined we now branch on each of the above branches with respect to the integer variables. If we obtain an integer solution which is *less* than any of the lower bounds on the discrete branches we can eliminate that discrete branch from further consideration. The evaluation continues until all branches have been evaluated or eliminated from consideration. To illustrate the method we consider the model for the optimal design of a vapor condensor given by Avriel and Wilde [1].

$$\min\ 172400\, N^{-7/6} D_0^{-1} L^{-4/3} + 97770\, D_i^{0.8} N^{-0.2} L^{-1} + 1.57\, N D_0 L + 0.0382\, L D_i^{-4.8} N^{-1.8} + 38380 (N D_i L)^{-1},$$

$$\text{subject to: } 0.00817 + D_i \leqq D_0,$$

where N = number of tubes (integer),
L = tube length (feet),
D_i = inside diameter (feet),
D_0 = outside diameter (feet).

The complete description of the model may be found in the above reference. For our purposes here we assume the tube has discrete outside diameters (D_0) of 1 inch, 0.75 inches and 0.5 inches.

We also assume the inside diameter (D_i) remains continuous in nature subject to the above constraint.

Upper and lower bounds were specified as:

$$\begin{aligned} 10 &\leqq N \leqq 800, \\ 10 &\leqq L \leqq 100, \\ 0.0001 &\leqq D_i \leqq 1, \\ 0.0001 &\leqq D_0 \leqq 1. \end{aligned} \tag{14}$$

Treating all variables as continuous the minimum cost design is found to be

N = 63.534 tubes,
L = 38.355 ft.,
D_i = 0.09877 ft.,
D_0 = 0.10694 ft.,
cost = \$896.963.

Constructing the discrete branches on the outside diameter we obtain the following designs where the costs define lower bounds on a particular branch:

	$D_0 = 1$ inch	$D_0 = 0.75$ inches	$D_0 = 0.5$ inches
N	111.792	218.091	585.667
L	28.083	19.331	11.009
D_i	0.07516	0.05433	0.03349
Cost	897.928	902.016	916.011
Iterations	28	23	21
Time	0.754 sec.	0.504 sec.	0.464 sec.

In order to obtain a design with an integer number of tubes we now construct branches on integer values of N. Since the cost associated with $D_0 = 1$ inch is the lowest we begin by branching there. The branches are $N \geqq 112$ and $N \leqq 111$ and at this point we begin to realize an additional advantage of the previously described cutting plane method. The branches can be incorporated directly in the above model in the form of additional constraints. The solution procedure then continues from the continuous solution (with respect to N) since a feasible point is not required, thus avoiding a complete restart of the algorithm.

	$D_0 = 1$ inch, $N \leqq 111$	$D_0 = 1$ inch, $N \geqq 112$
N	111 tubes	112
L	28.227 ft.	28.042 ft.
D_i	0.075163 ft.	0.075163 ft.
Cost	897.939	897.930
Additional iterations	9	17
Time	0.267	0.564

Since the cost associated with $N = 112$ is more attractive we conclude it is the optimal design. However, due to the difference with respect to $N = 111$, we might conclude we are indifferent between these designs based on cost. We remark it is *not necessary* to evaluate the remaining branches since the lower bounds on those branches are *higher* than the current integer solution. Thus while we had the potential of evaluating 6 branches we needed to evaluate 2 branches in order to determine the optimal solution. In terms of computational burden a total of 98 LP's were solved requiring 2.553 seconds of IBM 370/165 time to completely determine the optimal solution.

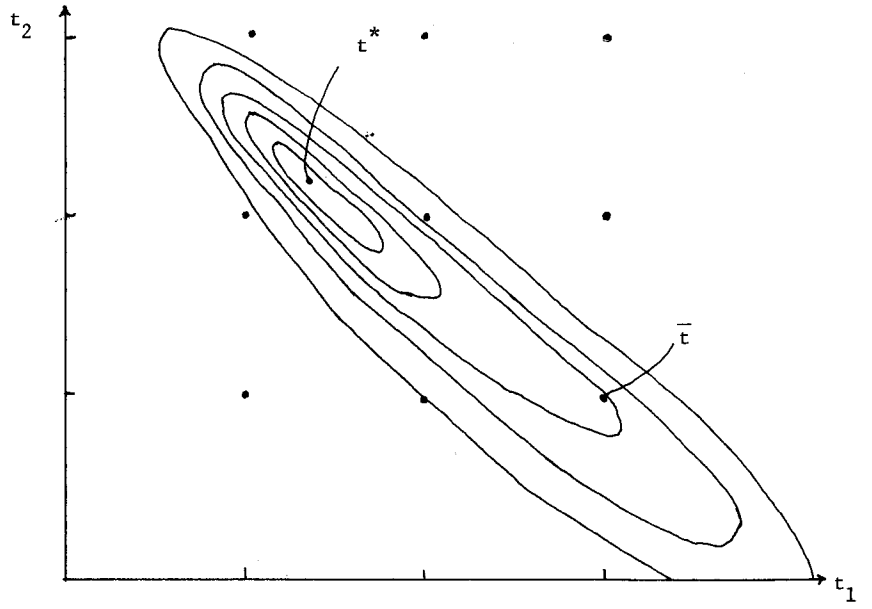

Figure 2.

We notice that in the above example the rounding of the continuous solution would have yielded the same result as obtained via the branch and bound method. However, as Figure 2 indicates this may not always be the case. The figure is to represent a hypothetical problem with two integer valued variables.

The point t^* is the optimal continuous solution and as can be seen from the objective function contours the nearest (with respect to t^*) integer solutions would not be optimal $\bar{t}$.

As pointed out by Avriel and Williams [3] not all design problems fit the restrictive form of posynomial geometric programs. This larger class of problems, termed polynomial or complementary geometric programs, does not exhibit the inherent convexity of the posynomial programs and as a result we can at best guarantee the attainment of a local optimal solution. Since the polynomial program can be reduced to a sequence of approximating posynomial programs [2, 3], the previously described algorithm can be used to obtain numerical solutions. Such a development has been given by Dembo *et al.* [6]. The combination of the branch and bound algorithm with the continuous solution procedure requires some modifications.

Following the development of Avriel and Williams [2, 3], we write *complementary geometric programs* as:

$$\begin{aligned} &\min t_0 , \\ &\text{subject to: } \frac{P_k(t)}{Q_k(t)} \leqq 1 , \quad k = 0, 1, \ldots, \rho , \\ &\qquad\qquad t = (t_0, \ldots, t_m) > 0 , \end{aligned} \tag{15}$$

where P_k and Q_k are posynomials.

Applying the geometric inequality

$$\sum_i u_i \geqq \prod_i (u_i/\varepsilon_i)^{\varepsilon_i} \tag{16}$$

to

$$Q_k(t) = \sum_{i=m_k}^{n_k} g_i(t) \text{ with } \varepsilon_i = \frac{g_i(\bar{t})}{Q_k(\bar{t})} \text{ for } \bar{t} > 0$$

we obtain the approximating program by replacing $Q_k(t)$ with the single term posynomial

$$\prod_{i=m_k}^{n_k} (g_i(t)/\varepsilon_i)^{\varepsilon_i} .$$

The algorithm given by Avriel and Williams [2, 3] generates a sequence of feasible points $t^{(k)}$ which are used to update the approximation and which under mild conditions converges to a local minimum [2]. The algorithm described in Section 2 can be used to solve this sequence of programs where the optimal solution to the approximating program is used to construct the next program in the sequence. The algorithm terminates when two points in the sequence satisfy some predetermined tolerance, i.e. $|t^{(k)} - t^{(k+1)}| < \text{TOL}$.

When applying the branch and bound technique to determine discrete solutions several considerations are important:

1. Since we may have obtained a local minimum as the continuous solution we cannot overlook the possibility of obtaining a discrete solution with a *lower* value. This, of course, would occur when we are in fact at a local solution and the branching technique forces us "closer" to the global solution. If the continuous solution is the global solution then its value does provide a lower bound on all feasible solutions. It is worthwhile to note that a modified Branch and Bound method has been proposed to generate the global continuous solution to this class of problems [10, 11].
2. We noted that when dealing with the integer branches such branches required only the addition of a constraint to the problem. For posynomial programs the algorithm of Section 2 does not require a feasible point, however, the Avriel and Williams approach to complementary programs requires the current point be feasible. Since the constraints defining the integer branches are violated we set up, according to the procedure of Dembo *et al.* [6], a penalized program

$$\begin{aligned} &\min Z_I , \\ &\text{s.t.} \frac{P_k(t)}{Q_k(t)} \leqq 1 , \quad k = 0, 1, \ldots, p , \\ &g_I(t) \leqq Z_I , \\ &Z_I \geqq 1 , \end{aligned} \tag{17}$$

where $g_I(t)$ is the added integer branch about the current non-integer solution.

This program is then solved using the algorithm of Section 2 and will yield a feasible solution to the original (unpenalized) program if $Z_I^* = 1$. Under this condition the optimal t values will provide a *feasible* point for the program associated with the integer branch. We should point out that since we are dealing with non-convex programs there may be no feasible solutions along such a branch; this situation will be identified by the solution to (16) being $Z_I^* > 1$.

4. Conclusions

The combination of the Branch and Bound technique and the cutting plane algorithm for ordinary geometric programs provides an efficient method for dealing with discrete valued non-linear problems. The method avoids the potential difficulties associated with simply rounding continuous solutions to discrete solutions and is readily implemented within a cutting plane algorithm. While the Branch and Bound could be used in connection with any optimization technique the cutting plane method is particularly well suited since the integer branches can be constructed with the addition of constraints and does not require a restart of the algorithm.

The combined method is applicable to the larger class of polynomial geometric programs following the approach of Avriel and Williams [2, 3].

REFERENCES

[1] M. Avriel and D. J. Wilde, Optimal Condenser Design by Geometric Programming, *Ind. Eng. Chem. Process Design and Dev.*, 6 (1967) 256–263.
[2] M. Avriel and A. C. Williams, Complementary Geometric Programming, *SIAM J. Appl. Math.*, 19 (1970) 125–141.
[3] M. Avriel and A. C. Williams, An Extension of Geometric Programming with Applications in Engineering Optimization, *J. of Eng. Math.*, 5 (1971) 187–194.
[4] E. Balas, A Note on the Branch and Bound Principle, *Oprs. Res.*, 16 (1968) 442–445.
[5] S. G. Beveridge and R. S. Schechter, *Optimization: Theory and Practice*, McGraw-Hill, New York (1970).
[6] R. Dembo, M. Avriel and U. Passy, *An Algorithm for the Solution of Generalized Geometric Programs*, paper presented at XIX TIMS, Houston, Texas, April 1972.
[7] J. J. Dinkel, W. H. Elliott and G. A. Kochenberger, *A Linear Programming Approach to Geometric Programs*, submitted for publication.
[8] R. J. Duffin, Linearizing Geometric Programs, *SIAM Review*, 12 (1970) 211–237.
[9] R. J. Duffin, E. L. Peterson and C. Zener, *Geometric Programming*, Wiley, New York (1967).
[10] J. E. Falk, *Global Solutions of Signomial Programs*, Report T-274, Institute for Management Science and Engineering, The George Washington University, June (1973).
[11] J. E. Falk and R. M. Soland, An Algorithm for Separable Non-Convex Programming Problems, *Management Science*, 15 (1969) 550–569.
[12] F. S. Hillier and G. J. Leiberman, *Introduction to Operations Research*, Holden-Day (1967).
[13] J. E. Kelley, The Cutting Plane Method for Solving Convex Programs, *SIAM J. Appl. Math.*, 8 (1960) 703–712.
[14] L. G. Mitten, Branch and Bound Methods: General Formulation and Properties, *Opers. Res.*, 18 (1970) 24–34.
[15] R. M. Stark and R. L. Nicholls, *Mathematical Foundations for Design, Civil Engineering Systems*, McGraw-Hill (1972).
[16] D. J. Wilde and C. S. Beightler, *Foundations of Optimization*, Prentice-Hall, Englewood Cliffs (1967).
[17] W. I. Zangwill, *Nonlinear Programming*, Prentice-Hall (1969).