No. SOC-49

WORST CASE NETWORK TOLERANCE OPTIMIZATION

J.W. Bandler, P.C. Liu and J.H.K. Chen

July 1974
(Revised January 1975)

WORST CASE NETWORK TOLERANCE OPTIMIZATION

John W. Bandler, Senior Member, IEEE,
Peter C. Liu, Student Member, IEEE,
and James H.K. Chen, Student Member, IEEE

Abstract

The theory and its implementation in a new user-oriented computer program package is described for solving continuous or discrete worst-case tolerance assignment problems simultaneously with the selection of the most favorable nominal design. Basically, the tolerance problem is to ensure that a design subject to specified tolerances will meet performance or other specifications. Our approach which is believed to be new to the microwave design area, can solve a variety of tolerance and related problems. Dakin's tree search, a new quasi-Newton minimization method and least pth approximation are used. The program itself is organized such that future additionsand deletions of performance specifications and constraints, replacement of cost functions and optimization methods are readily realized. Options and default values are used to enhance flexibility. The full Fortran listing of the program and documentation will be made available.

This work was supported by the National Research Council of Canada under Grant A 7239 and by a Scholarship to J.H.K. Chen.

J.W. Bandler and P.C. Liu are with the Group on Simulation, Optimization and Control and Department of Electrical Engineering, McMaster University, Hamilton, Canada.

J.H.K. Chen was with McMaster University. He is now with Bell-Northern Research, Ottawa, Canada.

I. INTRODUCTION

A new user-oriented computer program package called TOLOPT (TOLerance OPTimization) is presented which can solve continuous or discrete worst-case tolerance assignment problems simultaneously with the selection of the most favorable nominal design, taking full advantage of the most recent developments in optimization practice. Our approach, it is believed, is new to the microwave design area. Previous design work has usually been concentrated on obtaining a best nominal design, disregarding the manufacturing tolerances and material uncertainties. Basically, the tolerance assignment problem is to ensure that a design when fabricated will meet performance or other specifications.

The package is designed to handle the objective functions, performance specifications, and parameter constraints in a unified manner such that any of the nominal values or tolerances (relative or absolute) can be fixed or varied automatically at the user's discretion. Time-saving techniques for choosing constraints (vertices selection) are incorporated. The routine involved also checks assumptions and performs worst-case analyses. The paper also contains a brief discussion of network symmetry and how it may be implemented to further reduce the number of constraints.

The continuous and (optional) discrete optimization methods are programmed in such a way that they may be used as a separate unit. This part, called DISOP2 and incorporating several optional features, is an updated version of DISOPT, which has been successfully applied in many different areas [1] - [3]. Dakin's tree search for discrete problems [4], efficient gradient minimization of functions of many variables by a recent quasi-Newton method [5] and the latest developments in least pth approximation by Bandler and Charalambous [6] - [9] are employed. Extrapolation is also featured [10].

Another practical problem which is analogous to the tolerance assignment problem is to determine the optimum component values to a certain number of significant figures, which can be done with DISOP2.

The TOLOPT program is organized in such a way that future additions and deletions of performance specifications and constraints, replacement of cost functions and optimization methods are readily realized. Any of the two different vertices elimination schemes can be bypassed or replaced by the user. It is felt that the program is particularly flexible in the way that the user may enter at any stage of the problem's solution. The user supplies the network analysis subroutines. With an arbitrary initial acceptable or unacceptable design as a starting point, the program would output the set of nominal component parameters together with a set of optimal tolerances satisfying all the specifications in the worst-case sense. The user decides on a continuous solution and/or discrete solutions.

The package, written in Fortran IV, and run on a CDC 6400 digital computer will be made available. Several test examples are presented here to illustrate the theory and practice of the approach.

II. THE TOLERANCE PROBLEM

Introduction [11] - [15]

A design consists of design data of the nominal design point $\underset{\sim}{\phi}^{o} \overset{\Delta}{=} [\phi_1^o \; \phi_2^o \; \ldots \phi_k^o]^T$ and a set of associated tolerances $\underset{\sim}{\varepsilon} \overset{\Delta}{=} [\varepsilon_1 \; \varepsilon_2 \; \ldots \; \varepsilon_k]^T$, where k is the number of network parameters. Let $I_\phi \overset{\Delta}{=} \{1, 2, \ldots, k\}$ be the index set for these parameters. We take the ith absolute tolerance as ε_i in the discussion in this section, however, the discussion applies also to the relative tolerance $t_i \triangleq \frac{\varepsilon_i}{\phi_i^o}$, without any conceptual difference. An outcome of a circuit is any point $\underset{\sim}{\phi} \triangleq [\phi_1 \;\; \phi_2 \; \dots \; \phi_k]^T$ in the tolerance region $R_t \triangleq \{\underset{\sim}{\phi} | \phi_i^o - \varepsilon_i \leq \phi_i \leq \phi_i^o + \varepsilon_i,$ $i \in I_\phi\}$. The constraint region R_c is the region of points $\underset{\sim}{\phi}$ such that all performance specifications and constraints are satisfied by the circuit. The worst-case design requires that $R_t \subseteq R_c$. The optimal worst-case design can, therefore, be stated as: minimize some cost function C subject to $R_t \subseteq R_c$.

We need the following assumptions on R_c in order to make the problem tractable.

Assumptions on R_c

(1) R_c is not empty.

(2) R_c is bounded and simply connected.

(3) R_c is at least one-dimensionally convex.

Assumption (1) guarantees there is at least one feasible solution and (2) is a computational safeguard against infinite parameter values.

We say that R_c is one-dimensionally convex if for all $j \in I_\phi$ [11]

$$\underset{\sim}{\phi}^a, \; \underset{\sim}{\phi}^{b(j)} \triangleq \underset{\sim}{\phi}^a + \alpha \underset{\sim}{u}_j \in R_c \tag{1}$$

where α is some constant and $\underset{\sim}{u}_j$ is the jth unit vector, implies that

$$\underset{\sim}{\phi} = \underset{\sim}{\phi}^a + \lambda(\underset{\sim}{\phi}^{b(j)} - \underset{\sim}{\phi}^a) \in R_c \tag{2}$$

for all $0 \leq \lambda \leq 1$.

Let us also define the set of vertices $R_v \triangleq \{\underset{\sim}{\phi}^1, \underset{\sim}{\phi}^2, \dots, \underset{\sim}{\phi}^{2^k}\}$, and the corresponding index set I_v, where

$$\underset{\sim}{\phi}^r \triangleq \underset{\sim}{\phi}^o + \underset{\sim}{E}\underset{\sim}{\mu}(r) \tag{3}$$

$\mu_j(r) \in \{-1, 1\}$ and satisfies the relation

$$r = 1 + \sum_{j=1}^{k} \left(\frac{\mu_j(r) + 1}{2} \right) 2^{j-1} \tag{4}$$

$\underset{\sim}{E}$ is a diagonal matrix with ε_i as the ith element. Under the foregoing assumptions,

$$R_v \subseteq R_c \Rightarrow R_t \subseteq R_c \ . \tag{5}$$

See [11] for the proof, and Fig. 1 for an illustration of the concepts.

Assumptions on the Constraints

R_c may be defined specifically by a set of constraint functions, namely,

$$R_c \overset{\Delta}{=} \{\underset{\sim}{\phi} | g_i(\underset{\sim}{\phi}) \geq 0, \quad i \in I_c\} \tag{6}$$

where I_c is the index set for the functions. Concave constraint functions or, more generally, quasiconcave functions will satisfy assumption (3). The function $g(\underset{\sim}{\phi})$, (dropping the subscript i, $i \varepsilon I_c$), is said to be quasiconcave in a region if, for all $\underset{\sim}{\phi}^a$, $\underset{\sim}{\phi}^b$ in the region,

$$g(\underset{\sim}{\phi}^a + \lambda(\underset{\sim}{\phi}^b - \underset{\sim}{\phi}^a)) \geq \min[g(\underset{\sim}{\phi}^a), g(\underset{\sim}{\phi}^b)] \tag{7}$$

for all $0 \leq \lambda \leq 1$. An immediate consequence of (7) is that a region defined as $\{\underset{\sim}{\phi} | g(\underset{\sim}{\phi}) \geq 0\}$ is convex [16]. The intersection of convex regions is also convex and the multidimensional convexity implies the one-dimensional convexity of assumption (3).

If the point $\underset{\sim}{\phi}^b$ in (7) is defined as in (1), then, the function $g(\underset{\sim}{\phi})$ satisfying (7) will be called a one-dimensional quasiconcave function. The region defined by these functions is one-dimensionally convex. Assumption (3) is satisfied [17]. Throughout the following discussions, we will assume the functions to have this less restrictive property.

Under the foregoing assumptions we have the nonlinear programming problem:

minimize C subject to $g_i(\underset{\sim}{\phi}^r) \geq 0$ for all $\underset{\sim}{\phi}^r \in R_V$, $i \in I_c$.

Conditions for Monotonicity

Given a differentiable one-dimensional quasiconcave function $g(\phi)$, (see, for example, Fig. 2), the function is monotonic with respect to ϕ on an interval $[\phi^a, \phi^b]$ if $\text{sgn}\,(g'(\phi^a)) = \text{sgn}\,(g'(\phi^b))$. Furthermore, the minimum of $g(\phi)$ is at $\phi = \frac{1}{2}[\phi^a + \phi^b - \text{sgn}\,(g'(\phi^a))\,(\phi^b - \phi^a)]$. This may be proved as follows.

Consider the case $\text{sgn}\,(g'(\phi^a)) = \text{sgn}(g'(\phi^b)) > 0$. Suppose $g(\phi)$ is not monotonic. Then there exist two points ϕ^1, $\phi^2 \in (\phi^a, \phi^b)$, $\phi^2 > \phi^1$ such that $g'(\phi^1) < 0$ and $g(\phi^2) > g(\phi^1)$. Thus, $g(\phi^1 + \lambda(\phi^2 - \phi^1))$ for some $0 \leq \lambda \leq 1$ is smaller than $g(\phi^1)$ which contradicts (7). The assumption that $g(\phi)$ is not monotonic is wrong, hence, $g(\phi)$ is monotonic. Furthermore, it is nondecreasing on $[\phi^a, \phi^b]$. The minimum is at ϕ^a.

Similarly, it may be proved that the case $\text{sgn}(g'(\phi^a)) = \text{sgn}(g'(\phi^b)) < 0$ implies monotonicity with $g(\phi)$ nonincreasing on $[\phi^a, \phi^b]$. The minimum is at ϕ^b.

Implications of Monotonicity

Suppose g_i is monotonic in the same direction w.r.t. ϕ_j throughout R_t. Then the minimum of g_i is on the hyperplane $\phi_j = \phi_j^o - \varepsilon_j \text{ sgn } (\frac{\partial g_i}{\partial \phi_j})$. Hence, only those vertices which lie on that hyperplane need to be constrained. In general, if there are ℓ variables with respect to which the function g_i is monotonic in this way, the $2^{k-\ell}$ vertices lying on the intersection of the hyperplanes are constrained. In the case where $\ell = k$, the vertex of minimum g occurs at $\underset{\sim}{\phi}^r$, where

$$\phi_j^r = \phi_j^o - \varepsilon_j \text{ sgn } (\frac{\partial g_i}{\partial \phi_j}), \text{ for all } j \in I_\phi \quad . \qquad (8)$$

Let the set that contains the critical vertices be denoted by $R_V'(i) \subseteq R_V$. The modified problem is: minimize C subject to $g_i(\underset{\sim}{\phi}^r) \geq 0$, for all $\underset{\sim}{\phi}^r \in R_V'(i)$, $i \in I_c$.

The Vertices Elimination Schemes

Various schemes may be developed to identify or to predict the most critical vertices that are likely to give rise to active constraints. Our proposed schemes will eliminate all but one vertex for each constraint function in the most favourable conditions. In this case, the subsequent computational effort for the optimization procedure is comparable to the linearization technique commonly used. When monotonicity assumptions are not sufficient to describe the function behaviour, our scheme will increase the number of vertices until, at worst, all the 2^k vertices are included.

In principle, our schemes may be stated as follows:

Step (1): Systematic generation, for positive α, of sets of points

$$\underset{\sim}{\phi}^a, \ \underset{\sim}{\phi}^{b(j)} = \underset{\sim}{\phi}^a + \alpha \underset{\sim}{u}_j$$

Step (2): Evaluation of the function values and the partial derivatives at these points

Step (3): If $\operatorname{sgn}\left(\left.\frac{\partial g_i}{\partial \phi_j}\right|_{\underset{\sim}{\phi}=\underset{\sim}{\phi}^a} \right) = \operatorname{sgn}\left(\left.\frac{\partial g_i}{\partial \phi_j}\right|_{\underset{\sim}{\phi}=\underset{\sim}{\phi}^{b(j)}} \right)$

eliminate the vertices $\underset{\sim}{\phi}^r \varepsilon R_v$ on the hyperplane

$$\phi_j = \phi_j^o + \varepsilon_j \operatorname{sgn}\left(\frac{\partial g_i}{\partial \phi_j} \right)$$

If $\operatorname{sgn}\left(\left.\frac{\partial g_i}{\partial \phi_j}\right|_{\underset{\sim}{\phi}=\underset{\sim}{\phi}^a} \right) < 0$ and $\operatorname{sgn}\left(\left.\frac{\partial g_i}{\partial \phi_j}\right|_{\underset{\sim}{\phi}=\underset{\sim}{\phi}^{b(j)}} \right) > 0$ note that the quasiconcavity assumption is violated.

Comments

1. We have investigated and implemented two methods for step (1), involving

(a) $\underset{\sim}{\phi}^a = \underset{\sim}{\phi}^o - \varepsilon_j \underset{\sim}{u}_j$ and $\underset{\sim}{\phi}^b = \underset{\sim}{\phi}^o + \varepsilon_j \underset{\sim}{u}_j$, for all $j \ \varepsilon \ I_\phi$

(b) all the vertices of R_t.

A special case which we do not consider is for $\underset{\sim}{\phi}^a = \underset{\sim}{\phi}^b$ in step (1), in which case the first part of step (3) is applicable. $R_v'(i)$ contains only one vertex.

2. It is possible to further eliminate some vertices by considering the relative magnitudes of $g_i(\underset{\sim}{\phi}^r)$.
3. For method (b), a worst-case check can be accomplished as a by-product of the vertices elimination scheme since function values are computed at each vertex.
4. The schemes are based on local information. R_v' should be updated at suitable intervals.

<u>Symmetry</u>

A circuit designer should exploit symmetry to reduce computation time. The following is an example of how it may be done in the tolerance problem.

A function is said to be symmetrical w.r.t. $\underset{\sim}{S}$ in a region if

$$g(\underset{\sim}{S}\,\underset{\sim}{\phi}) = g(\underset{\sim}{\phi}) \tag{9}$$

where $\underset{\sim}{S}$ is a matrix obtained by interchanging suitable rows of a unit matrix [18]. It has exactly one entry of 1 in each row and in each column, all other entries being 0.

A common physical symmetry configuration is a mirror-image symmetry with respect to a center line. The $\underset{\sim}{S}$ matrix in this case is

$$\underset{\sim}{S} = \begin{bmatrix} \underset{\sim}{0} & & & 1 \\ & & 1 & \\ & \cdot^{\cdot^{\cdot}} & & \\ 1 & & & \underset{\sim}{0} \end{bmatrix} \tag{10}$$

Postmultiplication of a matrix $\underset{\sim}{A}$ by any $\underset{\sim}{S}$ simply permutes the columns of $\underset{\sim}{A}$ and premultiplication of $\underset{\sim}{A}$ permutes the rows of $\underset{\sim}{A}$. $\underset{\sim}{S}\underset{\sim}{S}^T = \underset{\sim}{1}$ and $\underset{\sim}{S}^T\underset{\sim}{D}\underset{\sim}{S} = \underset{\sim}{D}_s$, where $\underset{\sim}{D}$ is a diagonal matrix and $\underset{\sim}{D}_s$ is also a diagonal matrix with diagonal entries permuted.

Consider symmetrical $\underset{\sim}{S}$, $\underset{\sim}{\phi}^o$ and $\underset{\sim}{\varepsilon}$. By this we imply

$$\underset{\sim}{S}(\underset{\sim}{S}\,\underset{\sim}{A}) = \underset{\sim}{A} \quad , \tag{11}$$

$$\underset{\sim}{S}\,\underset{\sim}{\phi}^o = \underset{\sim}{\phi}^o \tag{12}$$

and

$$\underset{\sim}{S}^T \underset{\sim}{E} \underset{\sim}{S} = \underset{\sim}{E} \quad . \tag{13}$$

Let us premultiply the rth vertex from (3) by $\underset{\sim}{S}$, giving

$$\begin{aligned} \underset{\sim}{S} \underset{\sim}{\phi}^r &= \underset{\sim}{S} \underset{\sim}{\phi}^o + \underset{\sim}{S}(\underset{\sim}{E} \underset{\sim}{\mu}(r)), \; r \in I_V \\ &= \underset{\sim}{\phi}^o + \underset{\sim}{S}(\underset{\sim}{S}^T \underset{\sim}{E} \underset{\sim}{S} \underset{\sim}{\mu}(r)) \\ &= \underset{\sim}{\phi}^o + \underset{\sim}{E} \underset{\sim}{S} \underset{\sim}{\mu}(r). \end{aligned} \tag{14}$$

Now, $\underset{\sim}{S} \underset{\sim}{\mu}(r)$ is another vector with +1 and -1 entries. Let it be denoted by $\underset{\sim}{\mu}(s)$, $s \in I_V$. In some cases, $\underset{\sim}{\mu}(r)$ is identical to $\underset{\sim}{\mu}(s)$, if the vector is symmetrical. In other cases $\underset{\sim}{\mu}(r) \neq \underset{\sim}{\mu}(s)$. In all cases,

$$\underset{\sim}{S}\underset{\sim}{\phi}^r = \underset{\sim}{\phi}^s \quad . \tag{15}$$

Making use of the symmetrical property of g,

$$g(\underset{\sim}{S}\underset{\sim}{\phi}^r) = g(\underset{\sim}{\phi}^r) = g(\underset{\sim}{\phi}^s). \tag{16}$$

Let the number of symmetrical vectors $\underset{\sim}{\mu}(r)$ and the number of pairs of nonsymmetrical $\underset{\sim}{\mu}(r)$ and $\underset{\sim}{\mu}(s)$ be denoted by N(r=s) and N(r≠s), respectively.
Then

$$N(r=s) = 2^{k-k_s} \quad , \qquad 2k_s \leq k \tag{17}$$

and

$$N(r \neq s) = (2^k - 2^{k-k_s})/2 \quad , \qquad 2k_s \leq k \tag{18}$$

where k_s is the number of pairs of symmetrical components. There are, therefore, N(r=s)+N(r≠s) effective vertices as compared to 2^k topological vertices. Take, for example, k = 6 and k_s = 3. Only 36 function evaluations are required for all the 64 vertices.

The above discussion and results may be used to reduce computation time. However, in general, it is not certain that a nominal design without tolerances yielding a symmetrical solution will imply a symmetrical optimal solution with tolerances; either in the continuous or in the discrete cases.

III. OPTIMIZATION METHODS

Nonlinear Programming Problem

After eliminating the inactive vertices and constraints as discussed in Section II, the tolerance problem takes the form:

$$\text{minimize } f \overset{\Delta}{=} f(\underset{\sim}{x}) \tag{19}$$

subject to

$$g_i(\underset{\sim}{x}) \geq 0, \; i = 1,2,\ldots, m. \tag{20}$$

f is the chosen objective function (see Section IV). The vector $\underset{\sim}{x}$ represents a set of up to 2k design variables which include the nominal values, the relative and/or absolute tolerances of the network components. The constraint functions $g_1(\underset{\sim}{x})$, $g_2(\underset{\sim}{x})$, ..., $g_m(\underset{\sim}{x})$ comprise the selected response specifications, component bounds and any other constraints. The constraints are renumbered from 1 to m for simplicity.

Constraint Transformation

Recently, Bandler and Charalambous have proposed a minimax approach [8] to transform a nonlinear programming problem into an unconstrained objective. The method involves minimizing the function

$$V(\underset{\sim}{x},\alpha) = \max_{1 \leq i \leq m} \left[f(\underset{\sim}{x}),\; f(\underset{\sim}{x}) - \alpha g_i(\underset{\sim}{x})\right] \tag{21}$$

where

$$\alpha > 0 .$$

A sufficiently large value of α must be chosen to satisfy the inequality

$$\frac{1}{\alpha}\sum_{i=1}^{m} u_i < 1 \tag{22}$$

where the u_i's are the Kuhn-Tucker multipliers at the optimum. This approach compares favourably with the well-regarded Fiacco-McCormick technique [19].

Several least pth optimization algorithms are available for solving the resulting minimax problem. The function to be minimized is computed in the present paper as

$$U(\underset{\sim}{x}) \leftarrow (M(\underset{\sim}{x})-\varepsilon)\left(\sum_{j\varepsilon J}\left(\frac{e_j(\underset{\sim}{x}) - \varepsilon}{M(\underset{\sim}{x}) - \varepsilon}\right)^{q}\right)^{\frac{1}{q}} \tag{23}$$

where

$$M(\underset{\sim}{x}) \leftarrow \max_{j\varepsilon J} e_j(\underset{\sim}{x})$$

$$\varepsilon \leftarrow \begin{cases} 0 \text{ for } M(\underset{\sim}{x}) \neq 0 \\ \text{small positive number for } M(\underset{\sim}{x}) = 0 \end{cases}$$

$$q \leftarrow p \operatorname{sgn}(M(\underset{\sim}{x}) - \varepsilon)$$

$$p > 1$$

and

$$\text{if } M(\underset{\sim}{x}) \begin{cases} > 0,\ J \leftarrow \{j \mid e_j(\underset{\sim}{x}) > 0\} \\ < 0,\ J \leftarrow \{1,2,\ldots,m+1\} \end{cases}$$

The definition of the e_j's, the appropriate value(s) of p and the convergence features of the algorithms are summarized in Table I (algorithms 1 to 4).

Another approach to nonlinear programming which utilizes a least pth objective is also detailed in Table I (algorithm 5). It is a modification of an existing non-parametric exterior-point algorithm described by Lootsma [20].

Existence of a Feasible Solution

The existence of a feasible solution can be detected by minimizing (23) when

$$e_j \leftarrow \begin{cases} -g_i & , j = 1,2,\ldots,m \\ f - \bar{f} & , j = m+1 \end{cases}$$

where $\bar{f}$ is an upper bound on f. A nonpositive value of M at the minimum or even before the minimum is reached indicates that a feasible solution exists. Otherwise, no feasible solution satisfying the current set of constraints and the upper bound on the objective function value is perceivable. Only one single optimization with a small value of p greater than unity is required.

Unconstrained Minimization Method

Gradient unconstrained minimization methods have become very popular because of their reported efficiency. One such program is the Fortran subroutine, which utilizes first derivatives, implemented by Fletcher [5]. The method used belongs to the class of quasi-Newton methods. The direction of search $\underset{\sim}{s}^j$ at the jth iteration is calculated by solving the set of equations

$$\underset{\sim}{B}^j \underset{\sim}{s}^j = -\underset{\sim}{\nabla} U(\underset{\sim}{x}^j) \tag{24}$$

where $\underset{\sim}{B}^j$ is an approximation to the Hessian matrix $\underset{\sim}{G}$ of U, $\underset{\sim}{\nabla} U$ is the gradient vector and $\underset{\sim}{x}^j$ is the estimate of the solution at the jth iteration.

As proposed by Gill and Murray [21], the matrix $\underset{\sim}{B}^j$ is factorized as

$$\underset{\sim}{B}^j = \underset{\sim}{L}^j \underset{\sim}{D}^j \underset{\sim}{L}^{j^T} \tag{25}$$

where $\underset{\sim}{L}$ is a lower unit triangular matrix and $\underset{\sim}{D}$ a diagonal matrix. It is important that $\underset{\sim}{B}^j$ must always be kept positive definite and, with the above factorization, it is easy to guarantee this by ensuring $d_{ii} > 0$ for all i.

A modification of the earlier switching strategy of Fletcher [22] is used to determine the choice of the correction formula for the approximate Hessian matrix. The Davidon-Fletcher-Powell (DFP) formula is used if

$$\underset{\sim}{\delta}^T \underset{\sim}{L} \underset{\sim}{D} \underset{\sim}{L}^T \underset{\sim}{\delta} < \underset{\sim}{\delta}^T (\underset{\sim}{\nabla} U(\underset{\sim}{x}^{j+1}) - \underset{\sim}{\nabla} U(\underset{\sim}{x}^j))$$

where

$$\underset{\sim}{\delta} = \underset{\sim}{x}^{j+1} - \underset{\sim}{x}^j .$$

Otherwise, the complementary DFP formula is used.

The minimization will be terminated when $|x_i^{j+1} - x_i^j|$ is less than a prescribed small quantity, for all i.

Discrete Optimization

In practical design, a discrete solution may be more realistic than a continuous solution. In network tolerance optimization problems, very often only components of certain discrete values or having certain discrete tolerances are available on the market. At present, a general strategy for solving a nonlinear discrete programming problem is the tree-search algorithm due to Dakin [4].

Dakin's integer tree-search algorithm first finds a solution to the continuous problem. If this solution happens to be integral, the integer problem is solved. If it is not, then at least one of the integer variables, e.g., x_i, is non-integral and assumes a value x_i^*, say, in this solution. The range

$$[x_i^*] < x_i < [x_i^*] + 1$$

where $[x_i^*]$ is the largest integer value included in x_i^*, is inadmissible and consequently we may divide all solutions to the given problem into two non-overlapping groups, namely,

(1) solutions in which

$$x_i \leq [x_i^*]$$

(2) solutions in which

$$x_i \geq [x_i^*] + 1$$

Each of the constraints is added to the continuous problem sequentially and the corresponding augmented problems are solved. The procedure is repeated for each of the two solutions so obtained. Each resulting nonlinear programming problem thus constitutes a node and from each node two branches may emanate. A node will be fathomed if the following happens:

(1) the solution is integral

(2) no feasible solution for the current set of constraints is achievable

(3) the current optimum solution is worse than the best integer solution obtained so far.

The search stops when all the nodes are fathomed.

It seems, then, that the most efficient way of searching would be to branch, at each stage, from the node with the lowest $f(\underset{\sim}{x})$ value. This would minimize the searching of unlikely subtrees. To do this, all information about a node has to be retained for comparison and this may require cumbersome housekeeping and excessive storage for computer implementation. One way of compromising is to search the tree in an orderly manner; each branch is followed until it is fathomed.

The tree is not, in general, unique for a given problem. The tree structure depends on the order of partitioning on the integer variables used.

The amount of computation may be vastly different for different trees.

For the case of discrete programming problems subject to uniform quantization step sizes, the Dakin algorithm is modified as follows. Let x_i be the discrete variable which assumes a non-discrete solution, x_i^*, and q_i be the corresponding quantization step, then the two variable constraints added sequentially after each node become

$$x_i \geq [x_i^*/q_i]q_i + q_i$$

and

$$x_i \leq [x_i^*/q_i]q_i$$

The integer problem is thus a special case of the discrete problem with $q_i = 1$, $i = 1,2,\ldots,n$, where n is the number of discrete variables.

If, however, a finite set of discrete values given by

$$D_i = \{d_{i1}, d_{i2}, \ldots, d_{ij}, d_{i(j+1)}, \ldots, d_{iu}\},\ i = 1,2,\ldots,n$$

is imposed upon each of the discrete variables, the variable constraints are then added according to the following rules:

(1) if $d_{ij} < x_i^* < d_{i(j+1)}$, then add the two constraints

$$x_i \leq d_{ij}$$

and

$$x_i \geq d_{i(j+1)}$$

sequentially

(2) if $x_i^* < d_{i1}$, only add the constraint

$$x_i \geq d_{i1}$$

(3) if $x_i^* > d_{iu}$, only add the constraint

$$x_i \leq d_{iu}$$

The resulting nonlinear programming problem at each node is solved by one of the algorithms described earlier. The feasibility check is particularly useful here since the additional variable constraints may conflict with the original constraints on the continuous problem. An upper bound, $\bar{f}$, on $f(\underset{\sim}{x})$, if not specified, may be taken as the current best discrete solution. For a discrete problem, the best solution among all the discrete solutions given by letting variables assume combinations of the nearest upper and lower discrete values (if they exist) may be taken as the initial upper bound on $f(\underset{\sim}{x})$.

The new variable constraint added at each node excludes the preceding optimum point from the current solution space and the constraint is therefore active if the function is locally unimodal. Thus the value of the variable under the new constraint may be optionally fixed on the constraint boundary during the next optimization. See Fig. 3 for illustrations of the search procedure and a tree structure.

IV. IMPLEMENTATION OF THE TOLERANCE PROBLEM

The Overall Structure of TOLOPT

Fig. 4 displays a block diagram of the principal subprograms comprising the tolerance optimization program. A brief description of these subprograms is given in this section.

TOLOPT (TOLerance OPTimization program) is the subroutine called by the user. It organizes input data and coordinates other subprograms. Subroutine DISOP2 is a general program for continuous and discrete nonlinear programming problems. Subroutine VERTST eliminates the inactive vertices of the tolerance region. Subroutine CONSTR sets up the constraint functions based on the response specifications, component bounds and other constraints supplied in the user subroutine

USERCN. Subroutine COSTFN computes the cost function. The user has the option of supplying his own subroutine to define other cost functions. The user supplied subroutine NETWRK returns the network responses and the partial derivatives.

Table II is a summary of the features and options currently incorporated in TOLOPT.

Some components of $\underset{\sim}{\epsilon}$ and $\underset{\sim}{\phi}^o$ may be fixed which do not enter into the optimization parameters $\underset{\sim}{x}$. The user supplies the initial values of the tolerances (relative or absolute) and the nominals with an appropriate vector to indicate whether they are fixed or variable, relative or absolute. The program will assign those variable components to vector $\underset{\sim}{x}$.

The Objective Function

The objective function we have investigated and implemented is [11] - [13]

$$C = \sum_i \frac{c_i}{x_i} \tag{26}$$

where x_i is either ε_i or t_i and c_i are some suitable weighting factors supplied by the user. The default value is one. To avoid negative tolerances we let $x_i = x_i'^{\,2}$, where x_i' is taken as a new variable replacing x_i.

Vertices Selection and Constraints

Two schemes of increasing complexity are programmed in the subroutine. The user decides on the maximum allowable calls for each scheme, starting with the simple one. He may, if he wishes, bypass either one or even bypass the whole routine by supplying his own vertices or set up his own strategy of vertices selection routine.

The user supplies 3 sets of numbers, the elements of which correspond to the controlling parameter ψ_i, the specification S_i and the weighting factor w_i. ψ_i is an independent parameter, e.g., frequency, or any number to identify a particular function. w_i is given by

$$w_i = \begin{cases} +1 \text{ if } S_i \text{ is an upper specification} \\ -1 \text{ if } S_i \text{ is a lower specification.} \end{cases}$$

If both upper and lower specifications are assigned to one point, the user can treat it as two points, one with an upper specification and the other with a lower specification. The theory presented earlier will apply in this case under the monotonicity restrictions.

The scheme will, for each i select a set of appropriate $\underset{\sim}{\mu}$. Corresponding to each $\underset{\sim}{\mu}$, the values ψ_i, S_i and w_i are stored. This information is outputed, and used for forming the constraint functions.

The constraints associated with response specifications are of the form

$$g = w(S - F) \geq 0 \tag{27}$$

with appropiate subscripts, where F is the circuit response function of $\underset{\sim}{\phi}$ and ψ, and w and S are as before.

The parameter constraints are

$$\phi_j^o - \varepsilon_j - \phi_{\ell j} \geq 0 \tag{28}$$

and

$$\phi_{uj} - \phi_j^o - \varepsilon_j \geq 0 \tag{29}$$

where ϕ_{uj} and $\phi_{\ell j}$, $j \in I_\phi$ are the user supplied upper and lower bounds.

Updating Procedure

Once the constraints have been selected, optimization is started with a small value of p and α ($p = \alpha = 10$ as default values). We have decided to call the routine for updating constraints whenever the α value is updated or the optimization with the initial value of p is completed, until the maximum number of calls is exceeded or when there is no change of values for consecutive calls. For updating the values, we add new values of $\underset{\sim}{\mu}$ to the existing ones without any eliminations. This may not be the most efficient way but will be stable.

V. EXAMPLES

Example 1

To illustrate the basic ideas of different cost functions, variable nominal point, continuous and discrete solutions, a two-section 10:1 quarter-wave transformer is considered [23]. Table III shows the specifications of the design and the result of a minimax solution without tolerances. Fig. 5 shows the contours of $\max_i |\rho_i|$ over the range of sample points. The region R_c satisfies all the assumptions. Two cost functions, namely, $C_1 = \frac{1}{t_{Z_1}} + \frac{1}{t_{Z_2}}$ and $C_2 = \frac{1}{\varepsilon_{Z_1}} + \frac{1}{\varepsilon_{Z_2}}$ are optimized for the continuous case. The optimal solution with a fixed nominal point at $\underset{\sim}{a}$ yields a continuous tolerance set of 8.3% and 7.7% for C_1. For the same function with a variable nominal point, the set is {12.8, 12.8}% with nominal solution at $\underset{\sim}{b}$.

The tolerance set for C_2 is {15.0, 9.1}% with nominal solution at $\underset{\sim}{c}$. $\underset{\sim}{d}$ and $\underset{\sim}{e}$ correspond to the two discrete solutions with tolerance 10% and 15%. This example depicts an important fact that an optimal discrete solution cannot always be obtained by rounding or truncating the continuous tolerances to the discrete values. The nominal points must be allowed to move.

Example 2

To illustrate the branch and bound strategy, a 3-component LC lowpass filter is studied [12]. The circuit is shown in Fig. 6. Table IV summarizes the specifications and Table V lists the results for both the continuous and the discrete solutions. Two different tree structures are shown in Fig. 7 and Fig. 8. This example illustrates that the tree structure and hence the computational effort is dependent upon the order of partitioning on the discrete variables. A * attached to the node denotes an optimum discrete solution. It may be noted that one of the discrete solutions as well as the continuous solution yield symmetrical results although symmetry is not assumed in the formulation of the problem.

Example 3

Consider a 5-section cascaded transmission-line lowpass filter with characteristic impedances fixed at the values

$$Z_1^o=Z_3^o=Z_5^o=0.2$$

$$Z_2^o=Z_4^o=5.0$$

and terminated in unity resistances [1], [6]. See Table VI for the specifications.

The length units are normalized with respect to $\ell_q = c/4f_o$, where $f_o = 1$ GHz. Two problems are presented here.

1. A uniform 1% relative tolerance is allowed for each impedance. Maximize the absolute tolerances on the section lengths and find the corresponding nominal lengths. Let the cost function be

$$C_2 = \sum_{i=1}^{5} \frac{1}{\varepsilon_{\ell_i}} .$$

2. A uniform absolute length tolerance of .001 is given. Maximize the relative tolerances on the impedances and obtain the corresponding nominal lengths. Let the cost function be

$$C_1 = \sum_{i=1}^{5} \frac{1}{t_{Z_i}} .$$

The filter has 10 circuit parameters which may be arranged in the order $Z_1, Z_2, \ldots, Z_5, \ell_1, \ell_2 \ldots, \ell_5$. To simplify the problem, symmetry with respect to a center line through the circuit is assumed. The matrix $\underset{\sim}{S}$ is given by

$$\underset{\sim}{S} = \begin{bmatrix} \begin{matrix} & & & & 1 \\ & & & 1 & \\ & & 1 & & \\ & 1 & & & \\ 1 & & & & \end{matrix} & \underset{\sim}{0} \\ \underset{\sim}{0} & \begin{matrix} & & & & 1 \\ & & & 1 & \\ & & 1 & & \\ & 1 & & & \\ 1 & & & & \end{matrix} \end{bmatrix}$$

which also implies that $\ell_1^o = \ell_5^o$ and $\ell_2^o = \ell_4^o$. The same kind of equalities are applied to the tolerances.

The first vertices elimination scheme is applied with values at the optimal nominal values without tolerances and the relative impedance tolerance and the absolute length tolerances at 2% and .002, respectively. A total of 46 vertices corresponding to all the frequency points were selected from a possible set of 9×2^{10}. 14 were further eliminated by symmetry. A final total of 15 constraints were chosen after comparing relative magnitudes. These 15 constraints were used throughout the optimization. The continuous and discrete solutions to the two problems are shown in Table VII and Table VIII.

VI. DISCUSSION AND CONCLUSIONS

We have described an efficient user-oriented program for circuit design with worst-case tolerance considerations embodying a number of new ideas and recent algorithms. The automated scheme could start from an arbitrary initial acceptable or unacceptable design to obtain continuous and/or discrete optimal nominal parameter values and tolerances simultaneously. Optimization of the nominal values without tolerances should, however, preferably be done first to obtain a suitable starting point. The effort is small compared with the complete tolerance problem when a small value of p greater than unity, e.g., p=2, is used. An exact minimax solution is not needed. This also serves as a feasibility check. If R_c is indicated to be empty, the designer has to relax some specifications or change his circuit. The solution process may also provide valuable information to the designer, e.g., parameter or frequency symmetry.

The problem without tolerances may be solved easily by available programs such as CANOPT [24]. The user may alternatively utilize the optimization part, namely DISOP2, of the present package.

It is good practice to first obtain a continuous solution before attempting the discrete problem. A useful feature of the program is that, for example, depending on information obtained from prior runs, the user can re-enter at a number of different stages of the solution process.

The assumptions on the constraints may be difficult to test. For this reason, a Monte Carlo simulation of the final solution is usually carried out.

We have presented results for two basic types of cost function. A more realistic cost-tolerance model should be established from known component cost data if these are unsuitable in particular cases.

The complete Fortran listing and documentation for TOLOPT will be made available. It is very important that the user provided routine for network function computation and the respective sensitivities be efficient. Typical running time for a small and medium size problem (less than 10 network parameters or 20 optimization parameters) will be from 2 to 20 minutes. The execution time on a CDC 6400, taking the LC lowpass filter as an example, was less than 10 seconds for the continuous case and a total of 80 to 100 seconds for the entire problem, depending on the order of partitioning. The 5-section transmission-line example needed about 300 to 400 seconds.

REFERENCES

[1] J.W. Bandler, P.C. Liu and J.H.K. Chen, "Computer-aided tolerance optimization applied to microwave circuits", IEEE Int. Microwave Symp. Digest (Atlanta, Georgia, June 1974), pp. 275-277.

[2] J.W. Bandler, B.L. Bardakjian and J.H.K. Chen, "Design of recursive digital filters with optimum word length coefficients", 8th Princeton Conf. on Information Sciences and Systems (Princeton, N.J., March 1974).

[3] J.W. Bandler and J.H.K. Chen, "DISOPT - a general program for continuous and discrete nonlinear programming problems", Int. J. Systems Science, to be published. Also McMaster University, Hamilton, Canada, Internal Report in Simulation, Optimization and Control, No. SOC-29, March 1974 (full report by J.H.K. Chen).

[4] R.J. Dakin, "A tree-search algorithm for mixed integer programming problems", Computer J., vol. 8, 1966, pp. 250-255.

[5] R. Fletcher, "FORTRAN subroutines for minimization by quasi-Newton methods", Atomic Energy Research Establishment, Harwell, Berkshire, England, Report AERE-R7125, 1972.

[6] J.W. Bandler and C. Charalambous, "Practical least pth optimization of networks", IEEE Trans. Microwave Theory Tech., vol. MTT-20, Dec. 1972, pp. 834-840.

[7] C. Charalambous and J.W. Bandler, "New algorithms for network optimization", IEEE Trans. Microwave Theory Tech., vol. MTT-21, Dec. 1973, pp. 815-818.

[8] J.W. Bandler and C. Charalambous, "Nonlinear programming using minimax techniques", J. Optimization Theory and Applications, vol. 13, June 1974, pp. 607-619.

[9] C. Charalambous, "A unified review of optimization", IEEE Trans. Microwave Theory Tech., vol. MTT-22, March 1974, pp. 289-300.

[10] W.Y. Chu, "Extrapolation in least pth approximation and nonlinear programming", McMaster University, Hamilton, Canada, Internal Report in Simulation, Optimi-

zation and Control, No. SOC-71, Dec. 1974.

[11] J.W. Bandler, "Optimization of design tolerances using nonlinear programming", J. Optimization Theory and Applications, vol. 14, 1974, pp. 99-114.

[12] J.W. Bandler and P.C. Liu, "Automated network design with optimal tolerances", IEEE Trans. Circuits and Systems, vol. CAS-21, March 1974, pp. 219-222.

[13] J.F. Pinel and K.A. Roberts, "Tolerance assignment in linear networks using nonlinear programming", IEEE Trans. Circuit Theory, vol. CT-19, Sept. 1972, pp. 475-479.

[14] E.M. Butler, "Realistic design using large-change sensitivities and performance contours", IEEE Trans. Circuit Theory, vol. CT-18, Jan. 1971, pp. 58-66.

[15] B.J. Karafin, "The optimum assignment of component tolerances for electrical networks", B.S.T.J., vol. 50, April 1971, pp. 1225-1242.

[16] O.L. Mangasarian, Nonlinear Programming. New York: McGraw-Hill, 1969.

[17] J.W. Bandler and P.C. Liu, "Some implications of biquadratic functions in the tolerance problem", Proc. IEEE Int. Symp. Circuits and Systems (San Francisco, Calif. April 1974), pp. 740-744.

[18] P. Lancaster, Theory of Matrices. New York: Academic Press, 1969.

[19] A.V. Fiacco and G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques. New York: Wiley, 1968.

[20] F.A. Lootsma, "A survey of methods for solving constrained minimization problems via unconstrained minimization", in Numerical Methods for Nonlinear Optimization, F.A. Lootsma, Ed. New York: Academic Press, 1972.

[21] P.E. Gill and W. Murray, "Quasi-Newton methods for unconstrained optimization", J. Inst. Maths. and its Applications, vol. 9, 1972, pp. 91-108.

[22] R. Fletcher, "A new approach to variable metric algorithms", Computer J., vol. 13, 1970, pp. 317-322.

[23] J.W. Bandler and P.A. Macdonald, "Cascaded noncommensurate transmission-line networks as optimization problems", IEEE Trans. Circuit Theory, vol. CT-16, Aug. 1969, pp. 391-394.

[24] J.W. Bandler, J.R. Popović, and V.K. Jha, "Cascaded network optimization program", IEEE Trans. Microwave Theory Tech., vol. MTT-22, March 74, pp. 300-308.

TABLE I

THE OPTIONAL LEAST PTH ALGORITHMS

Algorithm	Definition of e_i	Convergence feature	Value(s) of p	Number of optimizations
1	$e_i \leftarrow \begin{cases} f - \alpha g_i, i=1,2,\ldots,m \\ f,\ i = m+1 \end{cases}$		Large	1
2	where $\alpha > 0$	Increment of p	Increasing	Implied by the sequence but superceded by the stopping quantity
3		Extrapolation	Geometrically increasing	
4	$e_i \leftarrow \begin{cases} f - \alpha g_i - \xi^r, i=1,2,\ldots,m \\ f - \xi^r, i = m+1 \end{cases}$ where $\alpha > 0$ $\xi^r \leftarrow \begin{cases} \min[0, M^0 + \gamma],\ r=1 \\ \check{M}^{r-1} + \gamma,\ r > 1 \end{cases}$ r indicates the optimization number; γ is a small positive quantity	Updating of ξ^r	Finite	Depend on the stopping quantity
5	$e_i \leftarrow \begin{cases} -g_i,\ i=1,2,\ldots,m \\ f - t^r,\ i = m+1 \end{cases}$ where $t^r \leftarrow \begin{cases} \text{optimistic estimate of } \check{f},\ r = 1 \\ t^{r-1} + \check{U}^{r-1},\ r > 1 \end{cases}$ r is defined as in 4	Updating of t^r		

TABLE II

SUMMARY OF FEATURES, OPTIONS, PARAMETERS AND SUBROUTINES REQUIRED

Features	Type	Options	Parameters[†]/subroutines
Design parameters	Nominal and tolerance	Variable or fixed Relative or absolute tolerances	Number of parameters Starting values Indication for fixed or variable parameters and relative or absolute tolerances
Objective function	Cost	Reciprocal of relative and/or absolute tolerances	Weighting factors
		Other	Subroutine to define the objective function and its partial derivatives
Vertices selection*	Gradient direction strategy		Maximum allowable number of calls of the vertices selection subroutine
Constraints	Specifications on functions of network parameters	Upper and/or lower	Sample points (e.g., frequency) Specifications Subroutine to calculate, for example the network response and its partial derivatives (NETWRK)
	Network parameter bounds		Upper and lower bounds
	Other constraints	As many as required	Subroutine to define the constraint functions and their partial derivatives (USERCN)

Nonlinear programming	Bandler-Charalambous minimax	Least pth optimization algorithms See Table I	Controlling parameter α Value(s) of p Test quantities for termination
	Exterior-point		Optimistic estimate of objective function Value of p
Solution feasibility check*	Least pth	Discrete problem Continuous and discrete problem	Constraint violation tolerance Value of p
Unconstrained minimization method	Quasi-Newton	Gradient checking at starting point by numerical perturbation	Number of function evaluations allowed Estimate of lower bound on least pth objective Test quantities for termination
Discrete optimization*	Dakin tree-search	Reduction of dimensionality User supplied or program determined initial upper bound on objective function Single or multiple optimum discrete solution Uniform or nonuniform quantization step sizes	Upper bound on objective function Maximum permissible number of nodes Discrete values on step sizes Number of discrete variables Discrete value tolerance Order of partitioning Indication for discrete variables

† Parameters associated with the options are not explicitly listed.

* These features are optional and may be bypassed.

TABLE III

TWO-SECTION 10:1 QUARTER-WAVE TRANSFORMER

Relative Bandwidth	Sample Points (GHz)	Reflection Coefficient Specification	Type
100%	0.5, 0.6, ..., 1.5	0.55	upper
Minimax solution (no tolerances) $|\rho| = 0.4286$			

TABLE IV

LC LOWPASS FILTER

Frequency Range (rad/s)	Sample Points (rad/s)	Insertion Loss Specification (dB)	Type
0 - 1	0.5, 0.55,0.6, 1.0	1.5	upper (passband)
2.5	2.5	25	lower (stopband)

Minimax solution (no tolerances)

passband 0.53 dB
stopband 26 dB

TABLE V

LC LOWPASS FILTER
TOLERANCE OPTIMIZATION (C_1)

Parameters	Continuous Solution		Discrete Solution From {1,2,5,10,15}%		
	Fixed Nominal	Variable Nominal	1	2	3
$x_2 = t_{L_1}$	3.5 %	9.9 %	5 %	10 %	10 %
$x_1 = t_C$	3.2 %	7.6 %	10 %	5 %	10 %
$x_3 = t_{L_2}$	3.5 %	9.9 %	10 %	10 %	5 %
$x_5 = L_1^o$	1.628		1.999		
$x_4 = C^o$	1.090		0.906		
$x_6 = L_2^o$	1.628		1.999		

TABLE VI

FIVE-SECTION TRANSMISSION-LINE LOWPASS FILTER

Frequency Range (GHz)	Sample Points (GHz)	Insertion Loss Specification (dB)	Type
0 - 1	.35,.4,.45,.75,.8,.85,1.0	.02	upper (passband)
2.5 - 10	2.5, 10	25	lower (stopband)

TABLE VII

FIVE-SECTION TRANSMISSION-LINE LOWPASS FILTER
TOLERANCE OPTIMIZATION (C_1)

Parameters	Continuous Solution	Discrete Solution From {.5, 1, 1.5, 2, 3, 5}%
$t_{Z_1} = t_{Z_5}$	3.56 %	3 %
$t_{Z_2} = t_{Z_4}$	2.27 %	2 %
t_{Z_3}	1.98 %	2 %
$\ell_1^o = \ell_5^o$	0.0786	
$\ell_2^o = \ell_4^o$	0.1415	
ℓ_3^o	0.1736	

$Z_1^o = Z_3^o = Z_5^o = 0.2, \quad Z_2^o = Z_4^o = 5$

$\varepsilon_{\ell_i} = 0.001, \quad i = 1,2,\ldots, 5$

TABLE VIII

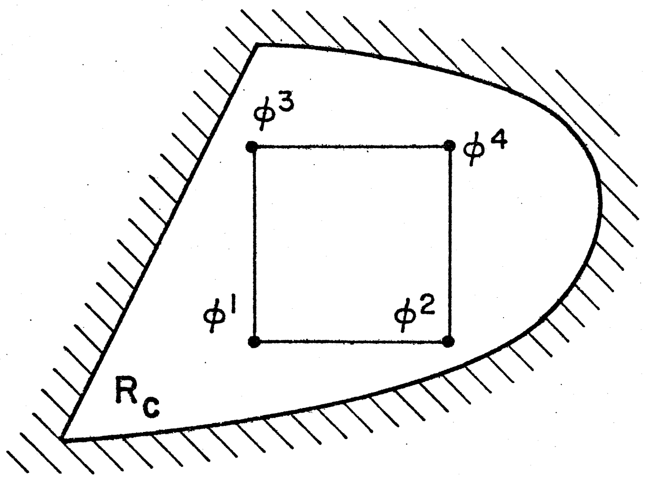
FIVE-SECTION TRANSMISSION-LINE LOWPASS FILTER
TOLERANCE OPTIMIZATION (C_2)

Parameters	Continuous Solution	Discrete Solution .0005 Step Size
$\varepsilon_{\ell_1} = \varepsilon_{\ell_5}$	0.0033	0.0030
$\varepsilon_{\ell_2} = \varepsilon_{\ell_4}$	0.0028	0.0030
ε_{ℓ_3}	0.0027	0.0025
$\ell_1^o = \ell_5^o$	0.0788	
$\ell_2^o = \ell_4^o$	0.1414	
ℓ_3^o	0.1738	

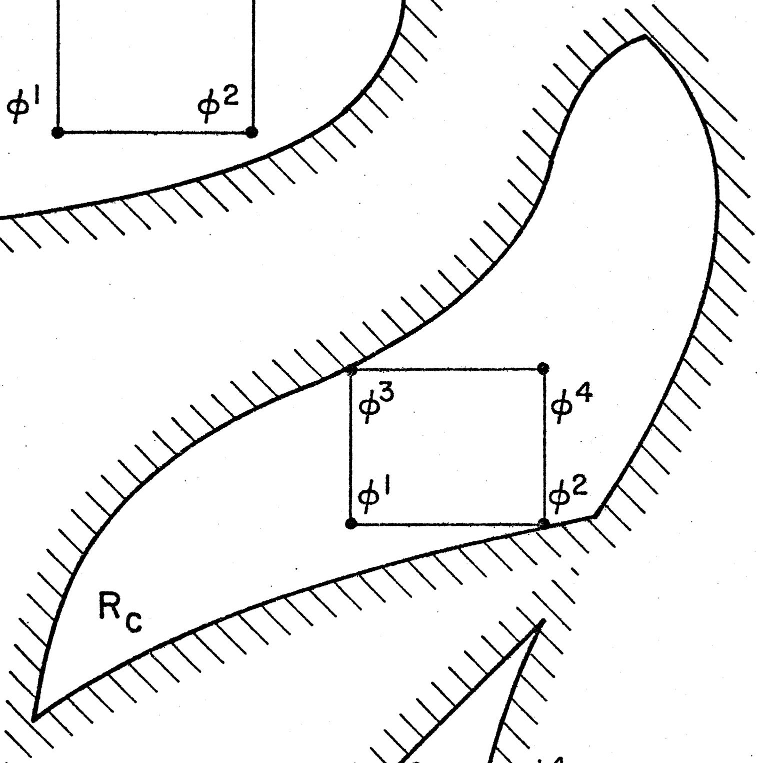
$Z_1^o = Z_3^o = Z_5^o = 0.2, \; Z_2^o = Z_4^o = 5$

$t_{Z_i} = 1\%, \quad i = 1,2,\ldots,5$

(a)



(b)



(c)

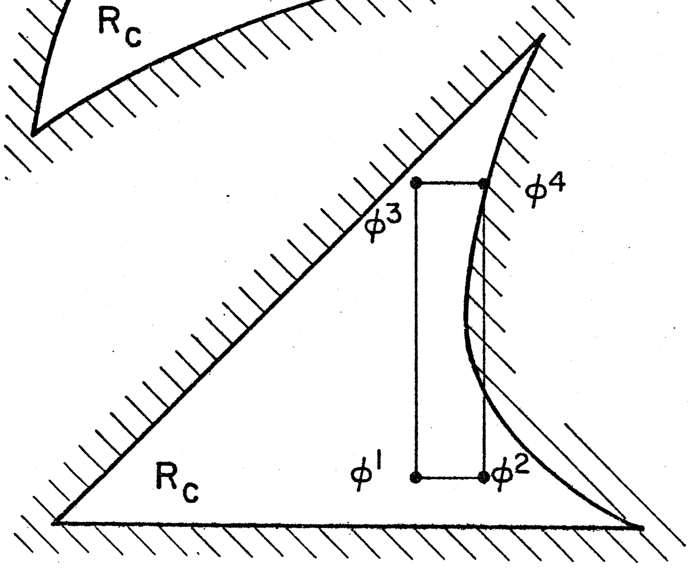


Fig. 1. Possible regions R_c.

(a) R_v is a subset of R_c implies that R_t is a subset of R_c.

(b) R_v is a subset of R_c implies that R_t is a subset of R_c.

(c) R_v is a subset of R_c does not imply that R_t is a subset of R_c.

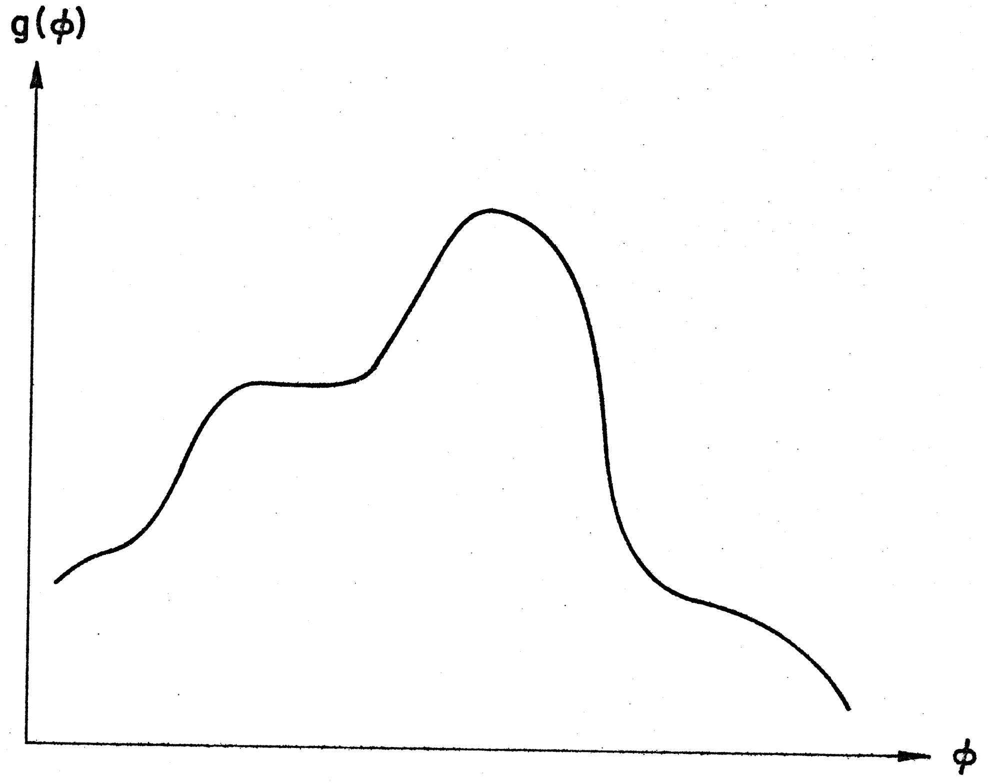


Fig. 2. A one-dimensional quasiconcave function.

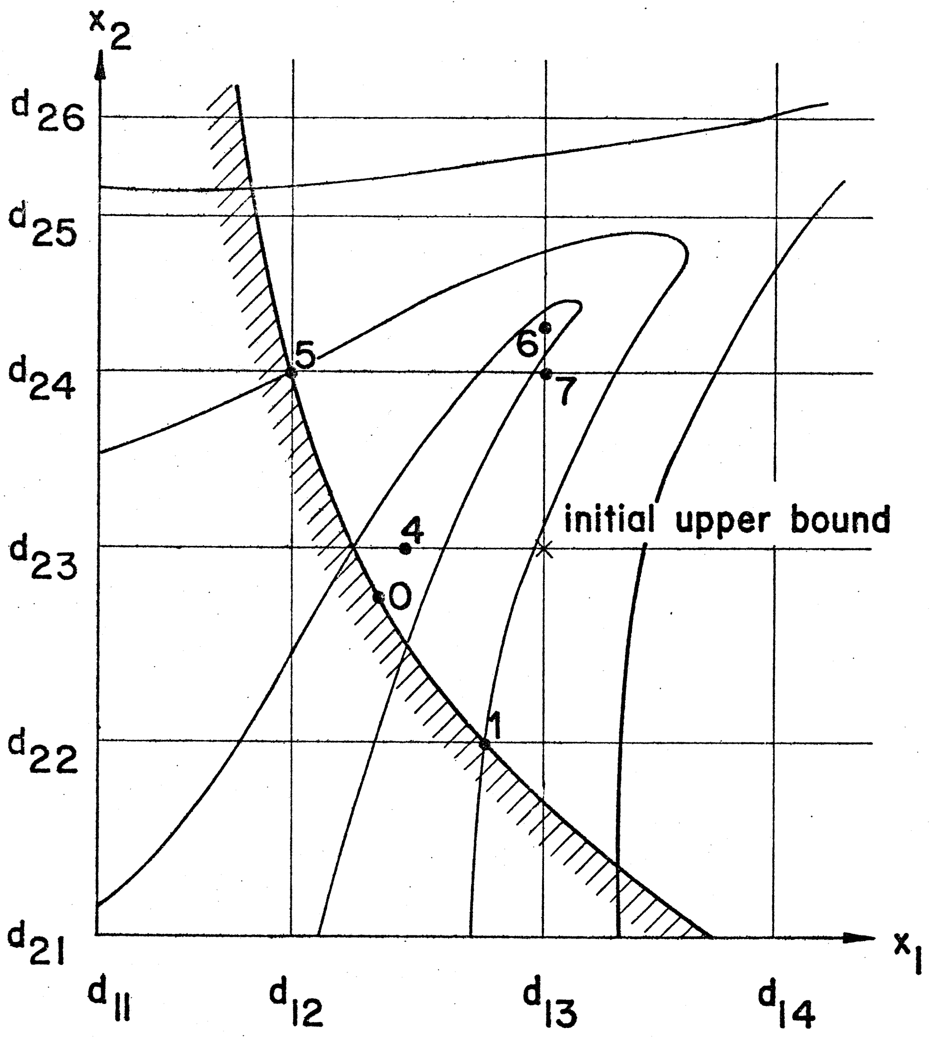
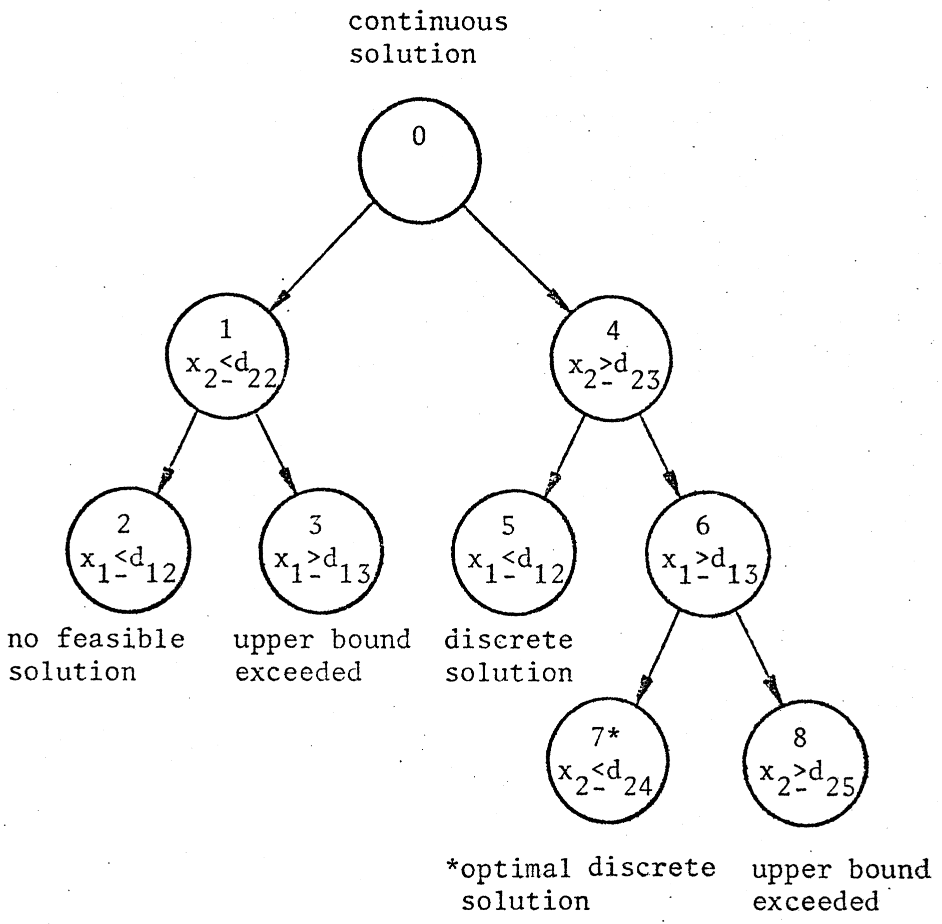


Fig. 3. An illustration of the search for discrete solutions.

(a) Contours of a function of two variables with grid and intermediate solutions.



(b) The tree structure.

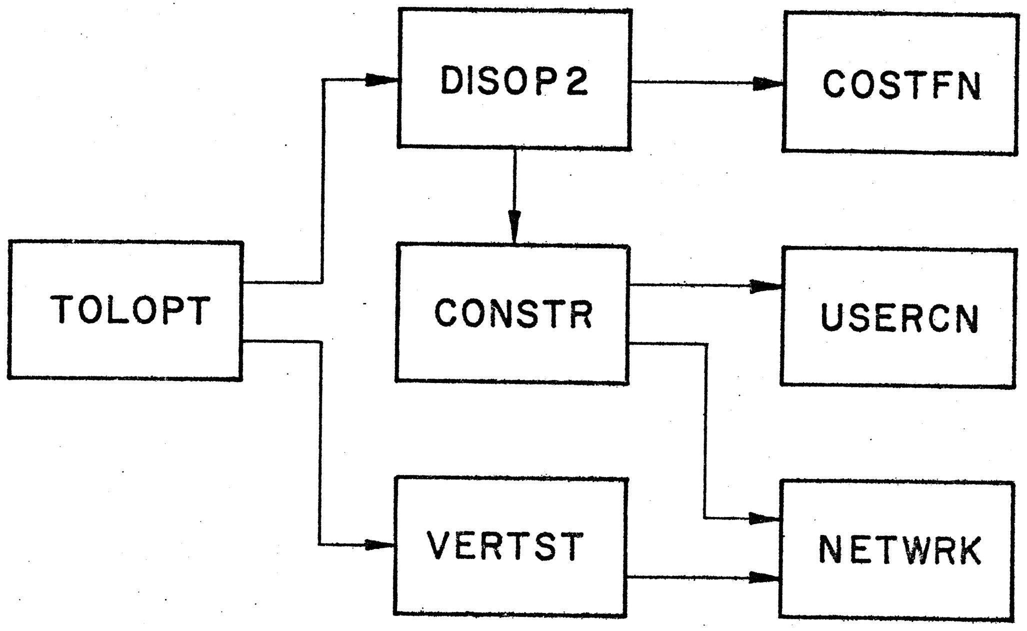


Fig. 4. The overall structure of TOLOPT. The user is responsible for NETWRK and USERCN.

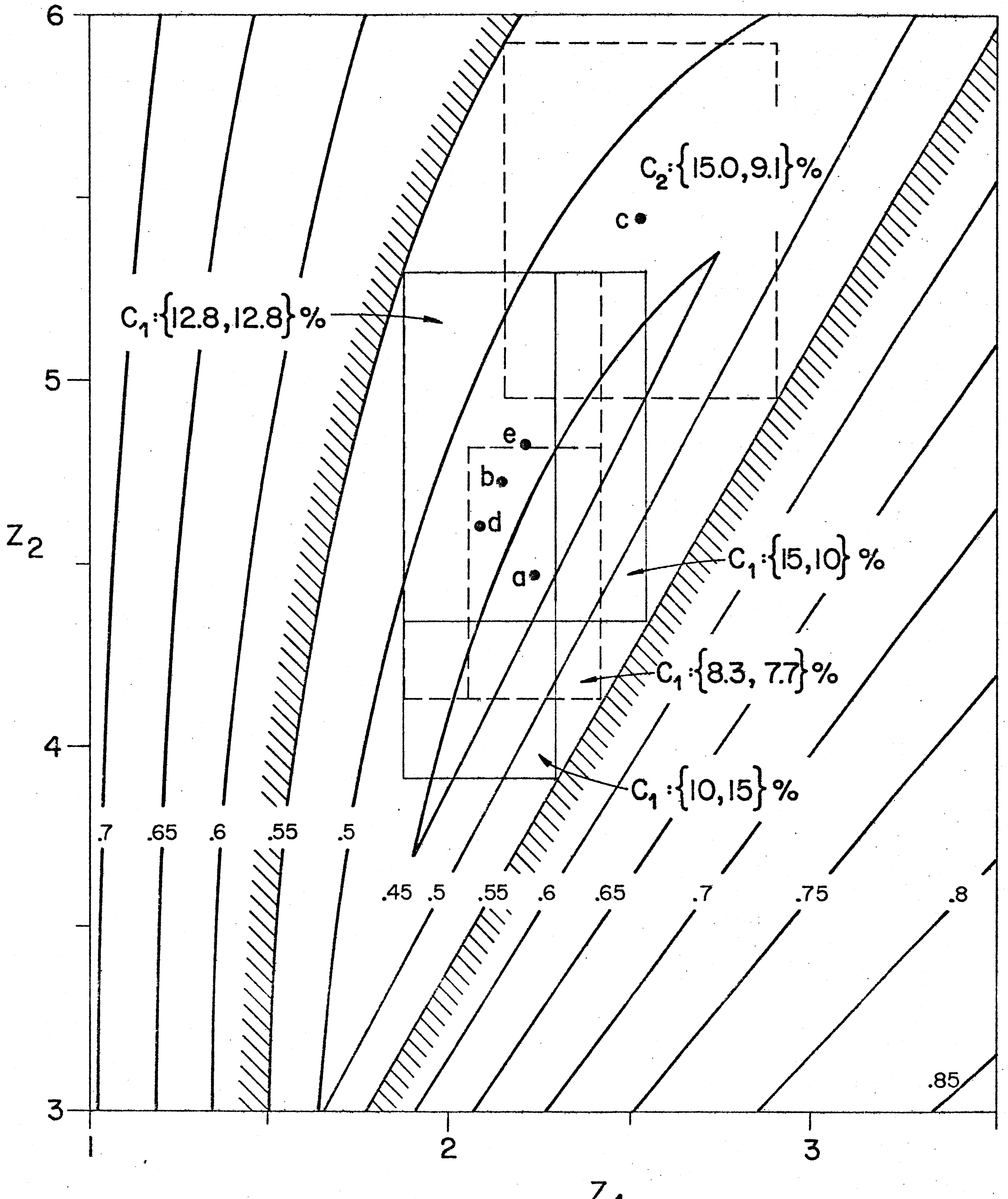


Fig. 5. Contours of max $|\rho_i|$ w.r.t. Z_1 and Z_2 for Example 1 indicating a number of relevant solution points (see text).

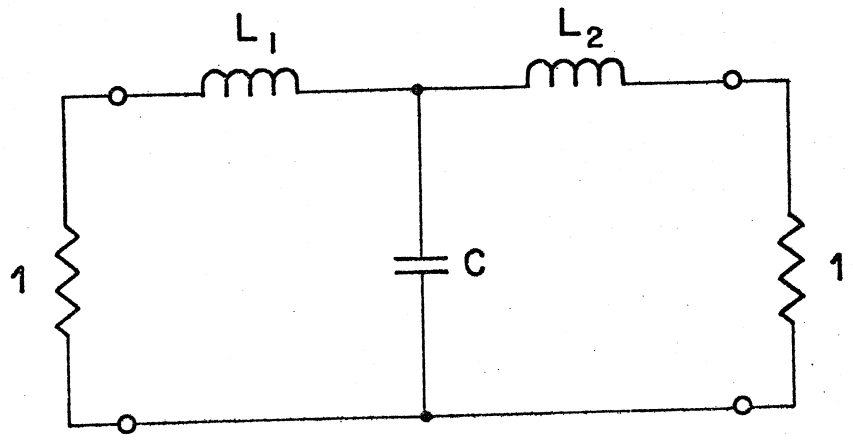


Fig. 6. The circuit for Example 2.

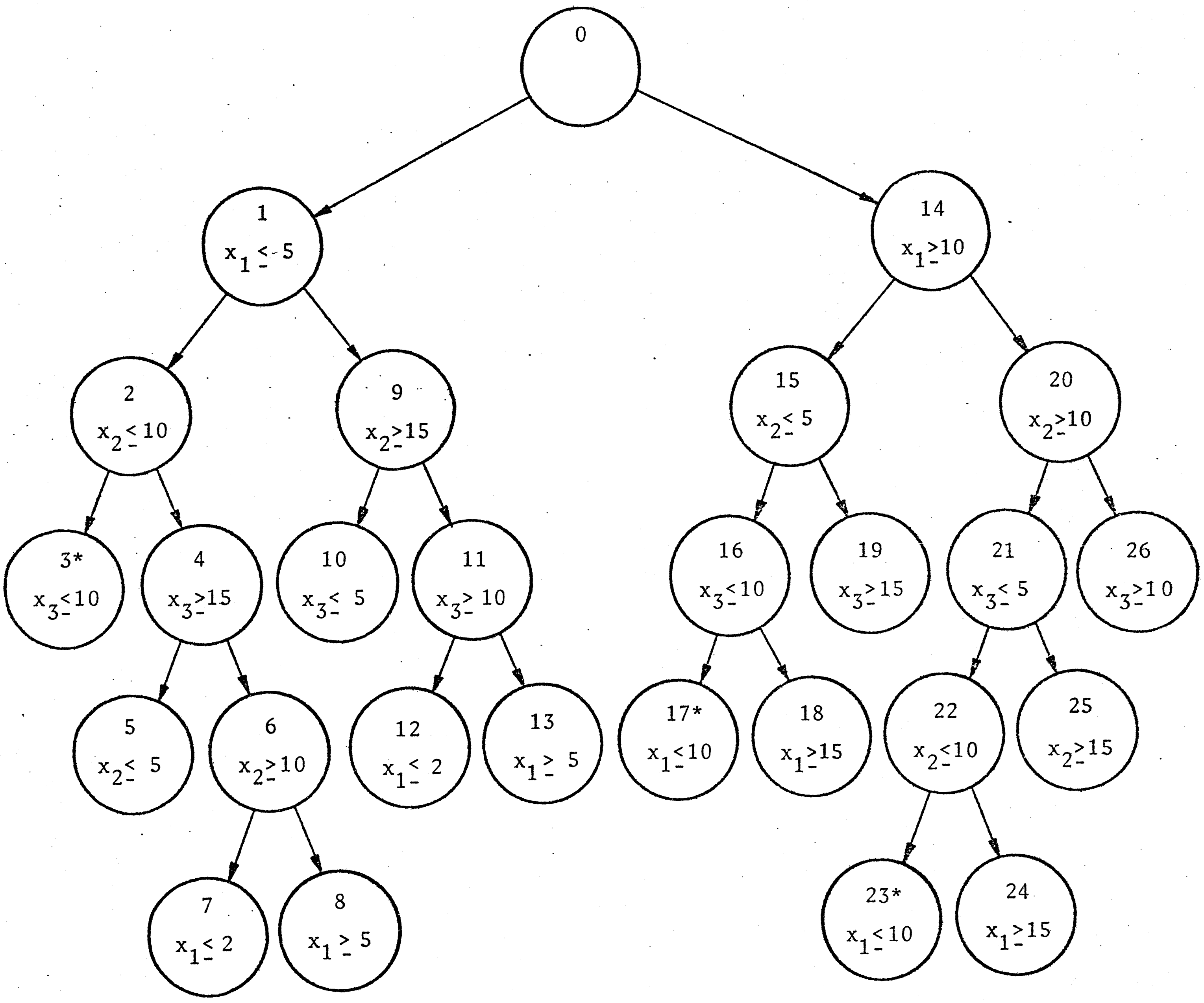


Fig. 7. Tree stucture for Example 2, partitioning on x_1 first (see Table V). *Denotes optimal discrete solutions.

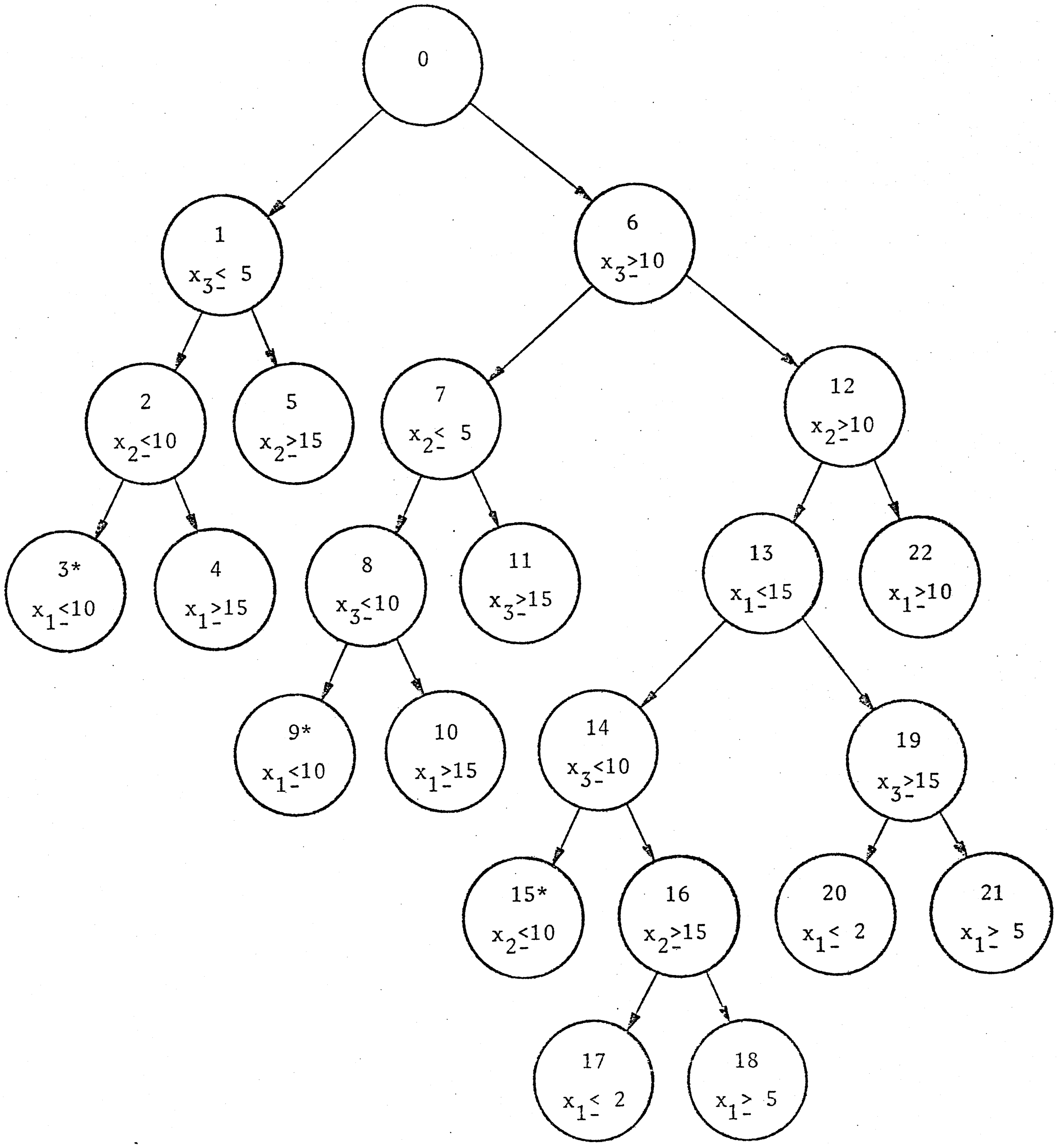


Fig. 8. Tree structure for Example 2, partitioning on x_3 first (see Table V). *Denotes optimal discrete solutions.