Proc. 21st Midwest Symposium on Circuits and Systems, Ames, Iowa, August 1978.

DISOPT3 - A USER-ORIENTED PACKAGE FOR NONLINEAR CONTINUOUS AND DISCRETE OPTIMIZATION PROBLEMS

J.W. Bandler and D. Sinha
Group on Simulation, Optimization and Control
and Department of Electrical Engineering
McMaster University, Hamilton, Canada L8S 4L7

Abstract

A package of FORTRAN subroutines called DISOPT3 for solving continuous and discrete, constrained or unconstrained general optimization problems is presented. The method used for arriving at the discrete solution involves conversion of the original constrained problem into a minimax problem by the Bandler-Charalambous technique, solving the continuous minimax problem using a recent least pth algorithm by Charalambous. Fletcher's 1972 method for unconstrained minimization and the Dakin branch and bound technique to generate the additional constraints.

1. INTRODUCTION

A package of FORTRAN subroutines called DISOPT3 for solving continuous and discrete, constrained or unconstrained general optimization problems is presented. The method used for arriving at the discrete solution involves conversion of the original constrained problem into a minimax problem by the Bandler-Charalambous technique [1], solving the continuous minimax problem using a recent least pth algorithm by Charalambous [2], Fletcher's method for unconstrained minimization [3] and the Dakin branch and bound technique [4] to generate the additional constraints. These steps are iteratively implemented until all the discrete solutions have been found. DISOPT3 is based conceptually on the DISOPT program developed by Bandler and Chen [5, 6]. All of the desirable features of DISOPT have been retained in DISOPT3 and some more have been added. DISOPT has been used as a yardstick against which the performance and validity of DISOPT3 have been measured. A CDC 6400 computer was used for developing and running this program.

The goal in developing DISOPT3 was to create an efficient user oriented program. This goal has been amply achieved. DISOPT3 not only incorporates some of the most efficient optimization algorithms but also conforms to the precepts of structured programming. For example, each subroutine performs only one function or some strongly related functions, the program listing is segmented into logical modules by means of comment cards, the use of GO TO statements is minimal, the logical structures are simple, and last but not least, descriptive comments are an integral part of the program listing enhancing its readability and ease of understanding.

The reader should consult, in addition to the references mentioned already [1-6], the following material dealing with the least pth approach in optimization: the paper by Bandler and Charalambous [7] introducing the least pth approach, some extensions [8-10] and a review article by Charalambous [11]. DISOPT3, following DISOPT, can solve, for example, recursive digital filter problems involving finite or optimal word lengths, analog circuit and system optimization including design centering and tolerance assignment (continuous or discrete). A number of problems from the literature have been employed to test the package [5,6,12,13].

2. BACKGROUND THEORY

2.1 ASSUMPTIONS

DISOPT3 may be used for solving a mixed continuous-discrete nonlinear programming problem which can be formulated as

$$\text{minimize } f(x_1, x_2, \ldots, x_N)$$

subject to

This work was supported by the National Research Council of Canada under Grant A7239.

$$g_1(x_1, x_2, \ldots, x_N) \geq 0 ,$$
$$g_2(x_1, x_2, \ldots, x_N) \geq 0 ,$$
$$\vdots$$

where $x_1, x_2, \ldots, x_K$ or $X(1), X(2), \ldots, X(K)$, where K is less than or equal to N, are variables that can vary continuously but must ultimately assume only certain specified values. These are called discrete variables. Out of the N variables, it is always the first K variables that we assume are discrete. There are two kinds of discrete variables. The first kind of variable can only take a finite number of values. The second kind of variable can assume values that correspond to uniformly spaced points on a line, i.e., any value belonging to the infinite set $(\ldots, -3a, -2a, -a, 0, a, 2a, 3a, \ldots)$ where a is a finite positive quantity. The number a may be called the step size of a uniformly discrete variable. Each of the $x_1, x_2, \ldots, x_K$ can be a discrete variable of either kind (but always the first K out of the N variables must be discrete).

2.2 GENERAL APPROACH

The nonlinear programming system presented above is solved assuming that all the variables can vary continuously. To obtain discrete values for certain variables in the optimal solution, further constraints are added to the problem and it is solved again, in an iterative manner. The flow diagram of Fig. 1 summarizes the technique.

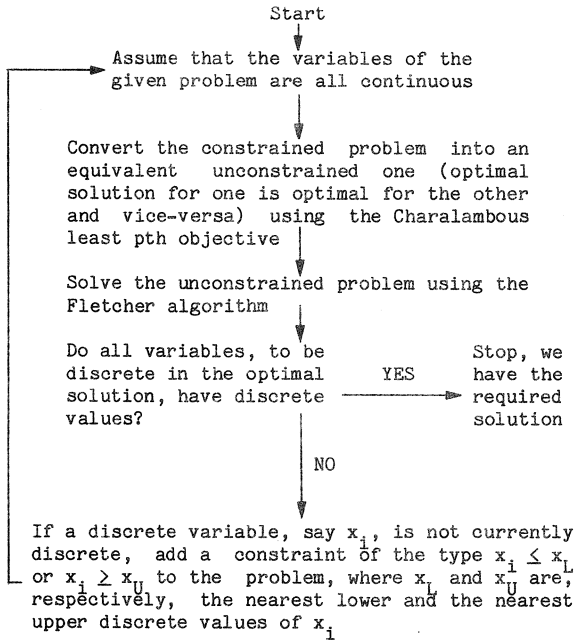

Fig. 1 General arrangement of DISOPT3.

2.3 LEAST pTH METHOD

For the nonlinear programming problem described earlier, Bandler and Charalambous have shown [1,2] that it is equivalent to the minimax problem

$$\min_{\underset{\sim}{x}} \max_{1\leq i\leq m} [f(\underset{\sim}{x}), f(\underset{\sim}{x}) - \alpha_i g_i(\underset{\sim}{x})]$$

where $\alpha_i > 0$ for $i = 1, 2, \ldots, m$, and sufficiently large.

Charalambous has stated the problem equivalently as

$$\min_{\underset{\sim}{x}} \max_{0\leq i\leq m} [f(\underset{\sim}{x}) - \alpha_i g_i(\underset{\sim}{x})]$$

where $\alpha_0 = 0$ and $\alpha_i > 0$ for $i = 1, 2, \ldots, m$, again as long as the α_i are above some threshold values.

This minimax formulation can now be converted, using the Bandler-Charalambous technique into the unconstrained least pth objective

$$U(\underset{\sim}{x},\underset{\sim}{\alpha},p,\xi) = M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi)\left[\sum_{i\in L(\underset{\sim}{x},\underset{\sim}{\alpha},\xi)}\left[\frac{f(\underset{\sim}{x})-\alpha_i g_i(\underset{\sim}{x})-\xi}{M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi)}\right]^q\right]^{\frac{1}{q}} \quad \text{for } M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi) \neq 0$$

$$= 0 \text{ for } M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi) = 0 ,$$

where

$$M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi) = \max_{i=0,1,\ldots,m} [f(\underset{\sim}{x})-\alpha_i g_i(\underset{\sim}{x})-\xi] ,$$

$$q = p \operatorname{sign} [M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi)] , \quad p > 1$$

$$L(\underset{\sim}{x},\underset{\sim}{\alpha},\xi) = \begin{cases} \{i \mid f(\underset{\sim}{x})-\alpha_i g_i(\underset{\sim}{x})-\xi>0\} & \text{if } M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi)>0 \\ I & \text{if } M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi) < 0 \end{cases}$$

and

$$I = 0, 1, \ldots, m .$$

The parameters p, $\underset{\sim}{\alpha}$ and ξ are fixed when minimizing U with respect to $\underset{\sim}{x}$.

The objective function U has the property that under the stated assumptions it is continuous with continuous first partial derivatives except when $M(\underset{\sim}{x},\underset{\sim}{\alpha},\xi) = 0$ and two or more of $(f(\underset{\sim}{x})-\alpha_i g_i(\underset{\sim}{x})-\xi)$ are 0 in which case U is continuous but the first partial derivatives are discontinuous.

Keeping p constant and changing ξ at each optimum point of U such that $U \to 0$ yields the desired minimax solution which, in turn, is the solution

of our original constrained problem. The key to the algorithm is the use of multipliers at each least pth solution to provide an estimate of the Kuhn-Tucker multipliers at the solution of the nonlinear program, which in turn are used in updating the $\underset{\sim}{\alpha}$. Charalambous derived a simple formula linking the multipliers at the solution with $\underset{\sim}{\alpha}$ such that, ideally, the minimum of U yields the desired solution independent of p and ξ.

2.4 BRANCH AND BOUND ALGORITHM

This algorithm starts by finding an optimal solution for the continuous problem. If this solution is discrete, then it is the required solution to the original problem. If the solution is nondiscrete, then at least one discrete variable, say x_i, lies in between x_L and x_U, the nearest lower and the nearest upper discrete values of x_i.

Since the range between x_L and x_U is inadmissible, we force all solutions into two subsets:

(i) solutions in which $x_i \leq x_L$

(ii) solutions in which $x_i \geq x_U$

The solution is analogous to a node generating another two nodes. The solution at each of the two new nodes includes one of the above constraints. The logical structure of this process, if continued, is that of a binary tree. The reader is referred to [4].

The tree search will always terminate in one of the three ways: we may reach a discrete solution or we may find that there is no feasible solution to the problem or the objective function at the current optimum solution is worse than the best discrete solution found so far. The solution to the original problem will be the best discrete solution found in this way. Hence, the problem is solved by searching this tree.

The tree is not, in general, unique for a given problem since at any stage we are at liberty to form the next constraint using any x_i which is nondiscrete; choosing different x_i will lead to different trees. More than one constraint may operate on a variable at one time.

For a discrete variable with uniform step size, x_L and x_U are determined as follows:

$$x_L = [x_i/a_i],$$
$$x_U = [x_i/a_i] + a_i.$$

3. COMPUTER PROGRAM DESCRIPTION

The programming package to solve the problem discussed in the preceeding sections is called DISOPT3. It contains FORTRAN subroutines. Some of its distinctive features can be summarized as follows:

(1) It is user oriented.

(2) It incorporates some of the most efficient optimization algorithms.

(3) It conforms to the precepts of structured programming. For example, each subroutine performs only one function or some strongly related functions, the program listing is segmented into logical modules by means of comment cards, the use of GO TO statements is minimal, the logical structures are simple, descriptive comments are an integral part of the program listing enhancing its ease of understanding, and the input and output variables for each subroutine have been clearly identified.

(4) The accompanying documentation [14] is so organized that it should be possible to solve problems after reading introductory Chapters 1 and 2. Chapter 3 has a discussion of the many available options. Chapter 4 deals with the concepts used in developing this program. Chapter 5 summarizes some results obtained by this program. The program listing is appended.

3.1 EXAMPLE 1: MODIFIED BANANA FUNCTION [5]

Minimize

$$f = 100((x_2+0.5) - (x_1+0.6)^2)^2 + (0.4 - x_1)^2,$$

where x_1 and x_2 are constrained to be natural numbers. The optimal solution is

$$f = 0.72$$
$$x_1 = 1.0$$
$$x_2 = 2.0$$

In order to arrive at this solution, many nodes are generated by the branch and bound algorithm. The solution at each node is shown in Table 1. The nodes are numbered to reflect the order in which they are generated.

TABLE 1
SUMMARY OF RESULTS FOR EXAMPLE 1

Node No.	Upper bound	Objective function	Solution x_1, x_2	Description
0	10^{10}	0	0.40, 0.50	continuous
1	2.12	0.16	0.00,-0.14	feasible
2	--	2.14	-0.56,-0.61	nonfeasible
3	--	2.12	0.00, 0.00	discrete
4	--	0.36	1.00, 2.06	feasible
5	--	0.72	1.00, 2.00	discrete
6	0.72	0.75	1.26, 2.99	nonfeasible

3.2 PROGRAM ORGANIZATION AND OPTIONS

There are eight subroutines in this program, in addition to the main program and subroutine FUN which are supplied by the user. Fig. 2 shows the overall organization for these subroutines. A → B

implies that subroutine B is called from subroutine A.

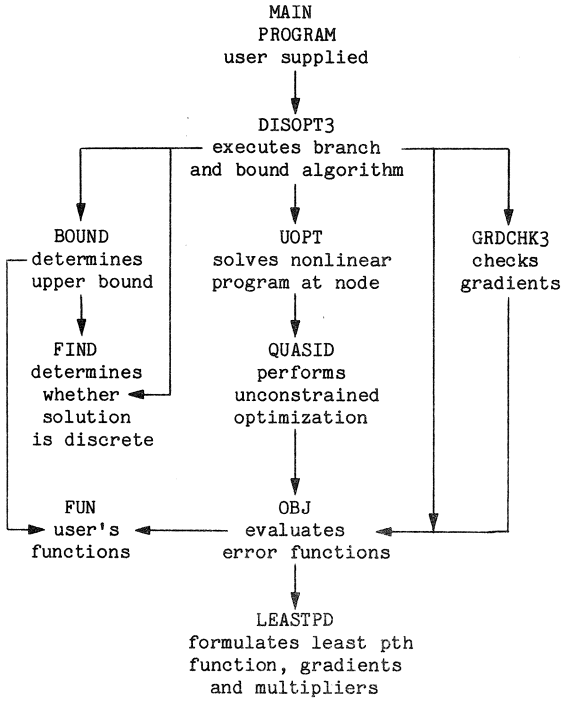

Fig. 2 Calling sequence for all the subroutines.

A useful feature of the program is that many of the variables in it have a default value; but it is possible and sometimes desirable to initialize these variables in the main program choosing different values. The user could, thus, opt for fast execution without a detailed printout or, a complete printout, etc. The performance of the program is greatly influenced by the choice of these values.

The many options available to the user through this program are, briefly, as follows.

3.2.1 One or All Discrete Solutions

If there are many optimal discrete solutions to a problem, will the user be satisfied with just one. If the answer is yes, let ONESOL, a logical variable, be TRUE; otherwise, FALSE. Finding all the solutions requires more effort than finding just one.

3.2.2 Checking Vertices for an Upper Bound

The effort required to find an optimal discrete solution using the branch and bound algorithm strongly depends on how soon a good upper bound can be found. If the user thinks that the objective function for his problem could not be larger than, say, 10.5 at the optimal discrete solution, he could set UPBND (the upper bound) = 10.5 in the main program. A value of UPBND which is lower than the actual objective function value (at the optimal discrete solution) will result in the program's inability to find any solution at all; whereas, too large a value will not save any effort.

An upper bound is automatically generated and updated whenever a discrete solution is found at a node but DISOPT3 also examines the discrete points surrounding the solution at node 0 if VERTCHK, a logical variable, is TRUE. This method of generating the upper bound could save a lot of effort if the user has no idea about the upper bound. If the user has a good idea, let VERTCHK be FALSE, and save some function evaluations.

3.2.3 Tolerances on Variables

The choice of numbers for such variables as TOLCONS, TOLDIS, TOLHEXI, TOLMULT and TOLX is critical to the efficiency of the program. All the tolerances should be chosen sufficiently small with respect to the magnitude of numbers involved in a problem. While too small a value for TOLX and TOLHEXI may result in excessive effort, too large a value could lead to the program's inability to find any solution at all. In test runs and to gain information about a problem, one could use large values and then switch to tight values along with some of the above features to economize on effort and obtain a more highly refined solution.

TOLCONS A small negative number. If a constraint value is smaller than 0 but larger than or equal to TOLCONS, it is considered as satisfied.

TOLDIS A small positive number. If a variable lies within TOLDIS neighbourhood of a discrete value, it is assumed to be discrete.

TOLHEXI A small positive number. Used by subroutine UOPT as a stopping criterion in the algorithm (see Charalambous [2]) that determines the continuous solution at each node.

TOLMULT A small positive number. Used in subroutine UOPT to select active constraints. If the multiplier (see Charalambous [2]) for a constraint exceeds TOLMULT, it is considered to be active. The active constraints are the only constraints that are used during the following optimization. By choosing TOLMULT as 0, the user can force all the constraints to be active all the time.

TOLX A small positive number. Used in subroutine QUASID (Fletcher algorithm [3]) to test the convergence of the solution.

3.2.4 Printing Options

Two hollerith variables, PRINTID and PRINTP, influence printing and offer the following options.

PRINTID = 3HYES — if the input data is to be printed, 2HNO otherwise.

PRINTP = 4HNONE — for no printing at all by any part of the program.

7HONLYDIS — for printing discrete solutions only.

7HNODEOPT — for printing the optimal solution at each node whether or not it is discrete.

3HALL — for printing the details of the optimization at each node. Results are printed after every IPT iterations of subroutine QUASID. IPT may also be changed by the user.

3.2.5 Check of the User's Gradients

Often, there is a mistake in the definition of gradients in subroutine FUN. The results obtained as such will be meaningless. This waste of effort might be avoided by setting GRADCHK, a logical variable, as TRUE.

When GRADCHK is true, the gradients are calculated (at the starting point) numerically and also by the user's definition. If the discrepancy is less than 10%, the user's definition is assumed to be correct; the possibility that the gradients are wrong must not still be ruled out, though. If the gradients are correct, a logical variable WRONG is FALSE; otherwise, it is TRUE and the program is terminated. In either case a message is printed.

3.2.6 Holding a Discrete Variable Constant

In the branch and bound algorithm, additional constraints, e.g., $X \leq XL$ or $X \geq XU$ are added to the problem if X is supposed to be a discrete variable but does not assume a discrete value in the optimal solution. There are two ways to implement it: (1) add the inequality constraint and optimize, (2) do not add the constraint, but hold X constant at the appropriate bound and optimize. The second alternative is, generally, more efficient and may be chosen by setting HOLDVAR, a logical variable, equal to TRUE. In the rare case when this method fails, it should not be used.

3.2.7 Branching on First or Last Variable

Many of the discrete variables may not have a discrete value in the solution. For the additional constraint, as explained above, should the first variable be chosen or the last? It is not possible to predict the best choice for every problem. However, if REVERSE, a logical variable, is TRUE the last variable is chosen.

3.2.8 Other Options

In addition to the variables described in the above options, the following could also be of interest to the user.

ALMIN — Used to initialize each element of vector AL. Vector AL is used to convert the nonlinear programming problem at each node into an exact minimax problem as proposed by Bandler and Charalambous [1]. ALMIN greatly influences the efficiency of the program but usually there is no way to predict a good value for a particular problem.

EST — An estimate of the optimal least pth function value at node 0. If initialized properly, this could save some function evaluations in the very first optimization.

IDCONS — An array identifying the active constraints, i.e., those constraints which are actually being used in the optimization at any node. This array may be used in subroutine FUN to evaluate only those constraints which are required.

IDVAR — An array identifying all the variables except the one which is held constant. If the evaluation of partial derivatives is very time consuming then IDVAR should be used in subroutine FUN to avoid the evaluation of those derivatives which are not needed.

IP — The parameter p of least pth optimization (see [2, 7-11]).

An exhaustive list and a complete description of the variables is provided in the program listing of subroutine DISOPT3 [14].

4. EXAMPLES

4.1 EXAMPLE 2: BEALE CONSTRAINED PROBLEM [5]

Minimize, as in the Beale problem [12],

$$f = 9 - 8x_1 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x_1x_3$$

subject to

$$x_1 \geq 0$$
$$x_2 \geq 0$$
$$x_3 \geq 0$$
$$3 - x_1 - x_2 - 2x_3 \geq 0$$

but where x_1, x_2 and x_3 are constrained to be natural numbers.

The optimal solutions are

$$f = 1.0$$
$$x_1 = 2.0 \quad x_1 = 1.0 \quad x_1 = 2.0$$
$$x_2 = 0.0 \quad x_2 = 1.0 \quad x_2 = 1.0$$
$$x_3 = 0.0 \quad x_3 = 0.0 \quad x_3 = 0.0$$

The results are summarized in Table 2.

TABLE 2
SUMMARY OF RESULTS FOR EXAMPLE 2

Node No.	Upper bound	Objective function	Solution x_1, x_2, x_3	Description
0	10^{10}	0.11	1.33,0.77,0.44	continuous
1	1.00	0.22	1.00,0.88,0.55	feasible
2	--	1.34	1.41,0.00,0.59	nonfeasible
3	--	0.25	1.00,1.00,0.50	feasible
4	--	1.00	1.00,1.00,0.00	discrete
5	--	1.07	0.32,0.91,1.00	nonfeasible
6	--	0.50	2.00,0.50,0.00	feasible
7	--	1.00	2.00,0.00,0.00	discrete
8	--	1.00	2.00,1.00,0.00	discrete

4.2 EXAMPLE 3: VOLTAGE DIVIDER PROBLEM [5,13]

Minimize

$$f = 1/x_1 + 1/x_2$$

subject to

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$0.53-(x_4+0.01x_2x_4)/(x_3-0.01x_1x_3+x_4+0.01x_2x_4) \geq 0$$

$$(x_4-0.01x_2x_4)/(x_3+0.01x_1x_3+x_4-0.01x_2x_4)-0.46 \geq 0$$

$$2.15 - x_4 - 0.01\, x_2\, x_4 - x_3 - 0.01\, x_1\, x_3 \geq 0$$

$$x_4 - 0.01\, x_2\, x_4 + x_3 - 0.01\, x_1\, x_3 - 1.85 \geq 0$$

where x_1 and x_2 both belong to the discrete set {1.0, 3.0, 5.0, 10.0, 15.0}.

The optimal solution is

$$f = 0.4$$
$$x_1 = 5.0$$
$$x_2 = 5.0$$
$$x_3 = 1.0130514$$
$$x_4 = 0.9901098$$

The results are summarized in Table 3.

4.3 PERFORMANCE WITH DIFFERENT OPTIONS

The performance of DISOPT3 contrasting the effects of different options is shown in Table 4 for Examples 1, 2 and 3.

5. CONCLUSIONS

An integrated computer program called DISOPT3 has been presented in this paper. Many of its features that make it a desirable program to use for solving continuous or discrete nonlinear programming problems have been discussed.

A fully documented 188 page report containing the complete listing and numerous examples is available at nominal charge [14].

TABLE 3
SUMMARY OF RESULTS FOR EXAMPLE 3

Node No.	Upper bound	Objective function	Solution $x_1,x_2,x3,x_4$	Description
0	10^{10}	0.28	7.00, 7.00, 1.01, 0.99	continuous
1	--	0.31	8.99, 5.00, 1.01, 0.99	feasible
2	--	0.40	5.00, 5.00, 1.01, 0.99	discrete
3	0.40	0.35	10.00, 3.99, 1.02, 0.99	feasible
4	--	0.41	12.29, 3.00, 1.01, 0.99	nonfeasible
5	--	0.30	10.00, 5.00, 1.01, 0.99	nonfeasible
6	--	0.35	3.99,10.00, 1.01, 0.99	feasible
7	--	0.41	3.00,12.43, 1.01, 0.99	nonfeasible
8	--	0.30	5.00,10.00, 1.01, 0.99	nonfeasible

TABLE 4
PERFORMANCE OF DISOPT3 WITH DIFFERENT OPTIONS

Feature	Number of function evaluations		
	Example 1	Example 2	Example 3
HOLDVAR = TRUE/FALSE	368/368	572/808	447/774
ONESOL = TRUE/FALSE	370/368	515/572	452/447
REVERSE = TRUE/FALSE	655/368	384/572	447/494
VERTCHK = TRUE/FALSE	368/581	572/788	447/447

REFERENCES

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

[2] C. Charalambous, "Nonlinear least pth optimization and nonlinear programming", Mathematical Programming, vol. 12, 1977, pp. 195-225.

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

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

[5] J.H.K. Chen, "DISOPT - a general program for continuous and discrete nonlinear programming problems", McMaster University, Hamilton, Canada, Report SOC-29, March 1974 (Revised June 1975).

[6] J.W. Bandler and J.H.K. Chen, "DISOPT - a general program for continuous and discrete nonlinear programming problems", Int. J. Systems Science, vol. 6, 1975, pp. 665-680.

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

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

[9] J.W. Bandler, C. Charalambous, J.H.K. Chen and W.Y. Chu, "New results in the least pth approach to minimax design", IEEE Trans. Microwave Theory Tech., vol. MTT-24, 1976, pp. 116-119.

[10] C. Charalambous and J.W. Bandler, "Nonlinear minimax optimization as a sequence of least pth optimization with finite valued of p", Int. J. Systems Science, vol. 7, 1976, pp. 377-391.

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

[12] J. Asaadi, "A computational comparison of some non-linear programs", Mathematical Programming, vol. 4, 1973, pp. 144-154.

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

[14] J.W. Bandler and D. Sinha, "DISOPT3 - a user-oriented package for nonlinear continuous and discrete optimization problems", McMaster University, Hamilton, Canada, Report SOC-174, July 1977.

John W. Bandler studied at Imperial College, London, England from 1960 to 1966. He received the B.Sc. (Eng.), Ph.D. and D.Sc. (Eng.) degrees from the University of London in 1963, 1967 and 1976, respectively. He joined Mullard Research Laboratories in England in 1966. From 1967 to 1969 he was a Postdoctoral Fellow and Sessional Lecturer at the University of Manitoba. He joined McMaster University in 1969, where he is currently Professor of Electrical Engineering and Chairman of the Department. Dr. Bandler is a member of the Association of Professional Engineers of the Province of Ontario and a Fellow of the IEEE.

Deepak Sinha obtained the B. Tech degree in Mechanical Engineering at I.I.T. Kanpur, India and received the M.S. degree in Industrial and Systems Engineering from the University of Florida, Gainesville, Florida in 1975. He joined the Group on Simulation, Optimization and Control at McMaster University in 1976 as a Research Assistant, where he developed computer software for optimization techniques. Presently he is employed with the McMaster University Computation Services Unit and involved in software development and conversion of large programs to run on small computers. Mr. Sinha is a member of the Association of Professional Engineers of the Province of Ontario.