INT. J. SYSTEMS SCI., 1975, VOL. 6, NO. 7, 665–680

DISOPT—A general programme for continuous and discrete non-linear programming problems†

J. W. BANDLER‡ and J. H. K. CHEN‡

An integrated computer programme in Fortran IV for continuous or discrete non-linear programming problems is presented. Several recent techniques and algorithms for non-linear programming have been adapted and new ideas have been introduced. They include the minimax and exterior-point approaches to non-linear programming, least pth optimization and the Dakin tree-search algorithm. The user may optionally choose the combination of techniques and algorithms best suited to his problems. Since many practical design problems can be easily formulated as non-linear programming problems, the programme, called DISOPT, enjoys a very wide range of applications such as continuous and discrete tolerance assignments, digital filter design, circuit design, system modelling and approximation problems. Numerical results for a number of functions and circuit tolerance optimization problems are presented in this paper to demonstrate the performance of DISOPT.

1. Introduction

Optimization has become an almost indispensable step in engineering design. Many useful algorithms and techniques for optimization have been proposed. However, it would be very time-consuming and inconvenient for each individual engineer to implement these algorithms and techniques to solve his particular design problem. The objective of this paper is to describe an efficient, user-oriented computer programme called DISOPT, which can solve continuous or discrete, constrained or unconstrained general optimization problems. Several recently proposed algorithms and techniques which have been reported to be efficient have been programmed into DISOPT. To the authors' knowledge, it is the first time that many of these algorithms and techniques are incorporated in a general programme. Several new ideas have also been introduced which allow the user to fully employ some of the latest developments.

Two approaches to non-linear programming are incorporated in DISOPT. The first is the minimax approach proposed by Bandler and Charalambous (1974), which compares favourably with the well-regarded sequential unconstrained minimization technique (Fiacco and McCormick 1968). For the implementation of this minimax approach, in addition to adapting the various least pth optimization algorithms due to Bandler and Charalambous, a new algorithm utilizing an extrapolation technique is developed. With all the attempted problems, this last algorithm was found to converge to the minimax optimum faster than the others. The

Received 4 November 1974.

† This work was supported by the National Research Council of Canada under Grant A7239 and by a Scholarship to J. H. K. Chen. This paper was presented at the Eighth Annual Princeton Conference on Information Sciences and Systems, Princeton, New Jersey., 28–29 March 1974.

‡ Group on Simulation, Optimization and Control, McMaster University, Hamilton, Canada.

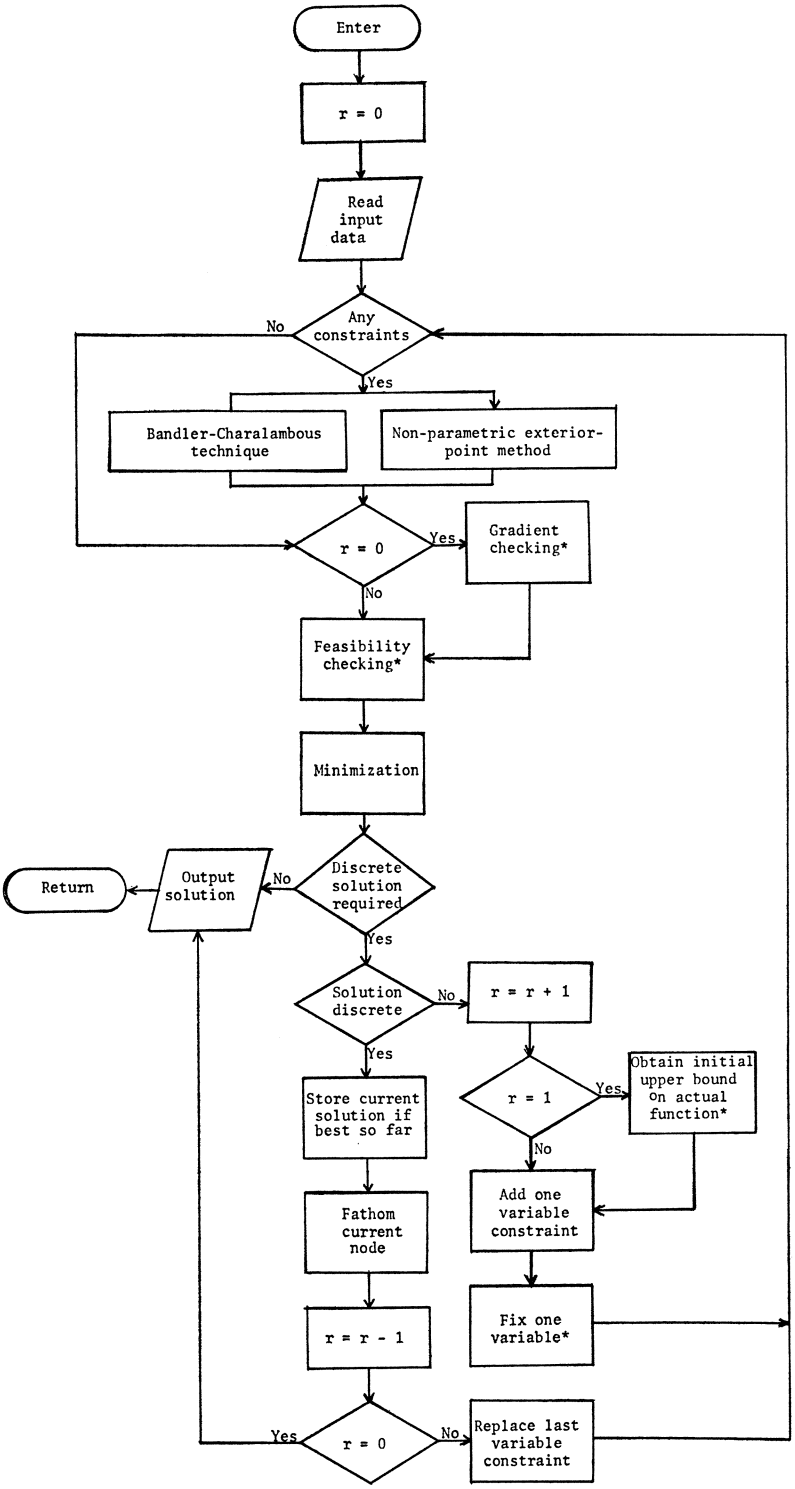

Figure 1. Flow diagram of DISOPT. * Indicates optional.

second approach to non-linear programming is a modification of an existing non-parametric exterior-point method described by Lootsma (1972). Some examples have been included to demonstrate the performance of the methods.

Recently, much attention has been directed to discrete optimization. The reason is that a discrete solution may be more realistic than a continuous solution. For example, in practical design problems, a compromise between maximum performance and minimum cost is often necessary because usually only components of certain discrete values are available on the market. Components of other values have to be custom-made and are therefore costly. The logic of the Dakin tree-search algorithm for integer programming (Dakin 1966) is followed but modifications have been embodied in DISOPT to enhance the efficiency of the algorithm. Some of them are :

(1) reduction of the dimensionality of the problem ;
(2) evaluation of an initial upper bound on the function value ;
(3) checking the existence of a feasible solution ; and
(4) determination of the availability of a better solution after a discrete solution is obtained.

The algorithm has also been generalized to handle discrete problems of uniform as well as non-uniform quantization step sizes. Several examples are given.

A programme due to Fletcher (1972) based on the work of Fletcher (1970) and Gill and Murray (1972) is employed to perform the minimization. The formulation of the required derivatives may be optionally checked by DISOPT using numerical perturbation. A flow diagram of the package is shown in Fig. 1.

2. The continuous optimization algorithms

Consider the non-linear programming problem of minimizing

$$f \triangleq f(\boldsymbol{\phi})$$

subject to

$$g_i(\boldsymbol{\phi}) \geqslant 0, \quad i = 1, 2, \ldots, m$$

where f is the objective function, the vector $\boldsymbol{\phi}$ represents a set of k variables

$$\boldsymbol{\phi} \triangleq [\phi_1 \ \phi_2 \ \cdots \ \phi_k]^{\mathrm{T}}$$

and $g_1(\boldsymbol{\phi}), g_2(\boldsymbol{\phi}), \ldots, g_m(\boldsymbol{\phi})$ are the constraint functions. Both f and the g_is are, in general, non-linear differentiable functions of the variables.

In order that efficient gradient minimization algorithms for unconstrained functions may be employed, the non-linear programming problem has to be transformed into an equivalent unconstrained objective. The two transformation methods used in DISOPT will be described next.

2.1. *Bandler–Charalambous technique* (1974)

The non-linear programming problem is transformed into the following unconstrained objective

$$V(\boldsymbol{\phi}, \alpha) = \max_{1 \leqslant i \leqslant m} [f(\boldsymbol{\phi}), f(\boldsymbol{\phi}) - \alpha g_i(\boldsymbol{\phi})]$$

where

$$\alpha > 0$$

Sufficiently large α must be chosen to satisfy the inequality

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

where the u_is are the Kuhn–Tucker multipliers at the optimum.

The minimization of $V(\boldsymbol{\phi}, \alpha)$ with respect to $\boldsymbol{\phi}$ is a minimax problem and may be implemented by one of the several recent least pth optimization algorithms proposed by Bandler and Charalambous (1972), Charalambous and Bandler (1973) and Charalambous (1974 a).

Let

$$e_i(\boldsymbol{\phi}) \triangleq f(\boldsymbol{\phi}) - \alpha g_i(\boldsymbol{\phi}), \quad i = 1, 2, \ldots, m$$

$$e_{m+1}(\boldsymbol{\phi}) \triangleq f(\boldsymbol{\phi})$$

$$M_e(\boldsymbol{\phi}) \triangleq \max_{1 \leqslant j \leqslant m+1} e_j(\boldsymbol{\phi})$$

Algorithm 1: Non-linear minimax optimization as a least pth optimization with a large value of p

Minimize with respect to $\boldsymbol{\phi}$ the function

$$U(\boldsymbol{\phi}) = (M_e(\boldsymbol{\phi}) - \epsilon)\left[\sum_{j \in J}\left(\frac{e_j(\boldsymbol{\phi}) - \epsilon}{M_e(\boldsymbol{\phi}) - \epsilon}\right)^q\right]^{1/q}$$

where

$$\epsilon = \begin{cases} 0 \text{ for } M_e(\boldsymbol{\phi}) \neq 0 \\ \text{small positive number for } M_e(\boldsymbol{\phi}) = 0 \end{cases}$$

$$q = p \text{ sign } (M_e(\boldsymbol{\phi}) - \epsilon)$$

and

$$\text{if } M_e(\boldsymbol{\phi}) \begin{cases} > 0, & 1 < p < \infty, \quad J = \{j \,|\, e_j(\boldsymbol{\phi}) > 0,\ j = 1, 2, \ldots, m+1\} \\ \leqslant 0, & 1 \leqslant p < \infty, \quad J = \{1, 2, \ldots, m+1\} \end{cases}$$

By employing a sufficiently large value of p, the minimization yields, for all practical purposes, a minimax solution.

Algorithm 2: Non-linear minimax optimization as a sequence of least pth optimization with increasing values of p

$U(\boldsymbol{\phi})$ is defined as in algorithm 1 and minimized using increasing values of p. The optimum of each minimization is used as the starting point of the following minimization. The process is terminated if the relative decrease in $M_e(\check{\boldsymbol{\phi}}^r)$, where $\check{\boldsymbol{\phi}}^r$ is the optimum of the rth minimization, between two consecutive minimizations is less than a preset small positive quantity or after p has reached the maximum assigned value.

Algorithm 3: Application of an extrapolation technique to a sequence of least pth optimizations with geometrically increasing values of p (Bandler and Chu 1974)

The basic formulation is the same as in the two previous algorithms. $U(\boldsymbol{\phi})$ is minimized with geometrically increasing values of p, i.e. $p^r = p^1 c^{r-1}$† where

† c^{r-1} means c raised to the power $r-1$.

p^r is the value of p used in the rth optimization and c is the multiplying factor. The optimum of each minimization is a function of p or $1/p$. The minimax solution is obtained as $p \to \infty$ or $1/p \to 0$.

Fiacco and McCormick (1968) have applied an extrapolation technique effectively to the SUMT constraint transformation algorithm. Since the situation here is analogous, it is felt that the convergence to the minimax solution may be improved by utilizing the same extrapolation technique. After each minimization, the extrapolation formula proposed by Fiacco and McCormick is applied to estimate the minimax optimum, $\check{\boldsymbol{\phi}}$, and the optimum of the next optimization.

Let $\boldsymbol{\phi}_j{}^i$, $i = 1, 2, \ldots, r$, $j = 0, 1, \ldots, i-1$ signify the jth order estimate of $\check{\boldsymbol{\phi}}$ after i minima have been achieved, then

$$\boldsymbol{\phi}_0{}^i = \check{\boldsymbol{\phi}}^i, \quad i = 1, 2, \ldots, r$$

where $\check{\boldsymbol{\phi}}^i$ is the optimum of the ith optimization and

$$\boldsymbol{\phi}_j{}^i = \frac{c^j \boldsymbol{\phi}_{j-1}{}^i - \boldsymbol{\phi}_{j-1}{}^{i-1}}{c^j - 1}, \quad i = 2, 3, \ldots, r, \; j = 1, 2, \ldots, i-1$$

The estimate of $\check{\boldsymbol{\phi}}$ is given by

$$\check{\boldsymbol{\phi}} = \boldsymbol{\phi}_{r-1}{}^r$$

To estimate $\check{\boldsymbol{\phi}}^{r+1}$, the recursive relation

$$\boldsymbol{\phi}_{j-1}{}^{r+1} = \frac{(c^j - 1)\boldsymbol{\phi}_j{}^{r+1} + \boldsymbol{\phi}_{j-1}{}^r}{c^j}$$

is used and

$$\check{\boldsymbol{\phi}}^{r+1} = \boldsymbol{\phi}_0{}^{r+1}$$

The process stops if the absolute difference between the estimate of $\check{\boldsymbol{\phi}}$ in two consecutive optimizations is less than a prescribed k-tuple $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k)$ where the elements are small positive numbers, or if the maximum allowable number of optimizations is exceeded.

Algorithm 4: Non-linear minimax optimization as a sequence of least pth optimization with finite values of p

(1) Define

$$\xi^1 = \min [0, M_e(\boldsymbol{\phi}^0) + \gamma]$$

where $\boldsymbol{\phi}^0$ is the starting point and γ is a small positive number.

(2) Set $r = 1$.

(3) Minimize with respect to $\boldsymbol{\phi}$ the function

$$U_\xi(\boldsymbol{\phi}, \xi^r) = (M_\xi(\boldsymbol{\phi}, \xi^r) - \epsilon) \left[\sum_{j \in J} \left(\frac{e_j(\boldsymbol{\phi}) - \xi^r - \epsilon}{M_\xi(\boldsymbol{\phi}, \xi^r) - \epsilon} \right)^q \right]^{1/q}$$

where

$$M_\xi(\boldsymbol{\phi}, \xi^r) = M_e(\boldsymbol{\phi}) - \xi^r$$

$$\epsilon = \begin{cases} 0 \text{ for } M_\xi(\boldsymbol{\phi}, \xi^r) \neq 0 \\ \text{small positive number for } M_\xi(\boldsymbol{\phi}, \xi^r) = 0 \end{cases}$$

$$q = p \text{ sign } M_\xi(\boldsymbol{\phi}, \xi^r)$$

and

$$\text{if } M_\xi(\boldsymbol{\phi}, \xi^r) \begin{cases} > 0, & \text{then } 1 < p < \infty, \quad J = \{j \,|\, e_j(\boldsymbol{\phi}) \geqslant \xi^r, \\ & \qquad\qquad\qquad\qquad j = 1, 2, \ldots, m+1\} \\ \leqslant 0, & \text{then } 1 \leqslant p < \infty, \quad J = \{1, 2, \ldots, m+1\} \end{cases}$$

(4) Set $\xi^{r+1} = M_e(\check{\boldsymbol{\phi}}^r) + \gamma$.
(5) If $|(\xi^{r+1} - \xi^r)/\xi^r| < \eta$, where η is a suitable small positive number, stop. Otherwise, set $r = r+1$.
(6) Go to step (3).

In this algorithm, any finite value of p greater than unity can be used to produce minimax solutions.

2.2. *A modified non-parametric exterior-point method*

This type of approach has been previously considered (Lootsma 1972, Charalambous and Bandler 1972, Charalambous 1974 b). In this work the non-linear programming problem is transformed into an equivalent least pth objective and implemented as follows.

Algorithm 5

(1) Let t^1 be the initial optimistic estimate of $f(\check{\boldsymbol{\phi}})$, i.e. $t^1 \leqslant f(\check{\boldsymbol{\phi}})$.
(2) Set $r = 1$.
(3) Minimize with respect to $\boldsymbol{\phi}$, the function

$$U_t(\boldsymbol{\phi}, t^r) = M_t(\boldsymbol{\phi}, t^r) \left[\left(\frac{f(\boldsymbol{\phi}) - t^r}{M_t(\boldsymbol{\phi}, t^r)} \right)^p + \sum_{j \in J} \left(\frac{-g_j(\boldsymbol{\phi})}{M_t(\boldsymbol{\phi}, t^r)} \right)^p \right]^{1/p}$$

where

$$M_t(\boldsymbol{\phi}, t^r) = \max_{j \in J} [f(\boldsymbol{\phi}) - t^r, \; -g_j(\boldsymbol{\phi})]$$

$$J = \{j \,|\, g_j < 0\}$$

and

$$1 < p < \infty$$

(4) Set $t^{r+1} = t^r + U_t(\check{\boldsymbol{\phi}}^r, t^r)$.
(5) If $|(t^{r+1} - t^r)/t^r| < \eta$, where η is a small positive number, stop. Otherwise, set $r = r+1$.
(6) Go to step (3).

Theorem 1

If $t^r \leqslant f(\check{\boldsymbol{\phi}})$, then $t^{r+1} \leqslant f(\check{\boldsymbol{\phi}})$.

Proof

By definition of $\check{\boldsymbol{\phi}}^r$,

$$U_t(\check{\boldsymbol{\phi}}^r, t^r) \leqslant U_t(\check{\boldsymbol{\phi}}, t^r) = M_t(\check{\boldsymbol{\phi}}, t^r) \left[\left(\frac{f(\check{\boldsymbol{\phi}}) - t^r}{M_t(\check{\boldsymbol{\phi}}, t^r)} \right)^p + \sum_{j \in J} \left(\frac{-g_j(\check{\boldsymbol{\phi}})}{M_t(\check{\boldsymbol{\phi}}, t^r)} \right)^p \right]^{1/p}$$

$$= M_t(\check{\boldsymbol{\phi}}, t^r) \left[\left(\frac{f(\check{\boldsymbol{\phi}}) - t^r}{M_t(\check{\boldsymbol{\phi}}, t^r)} \right)^p \right]^{1/p}$$

(since $g_i(\check{\boldsymbol{\phi}}) \geqslant 0$, $i = 1, 2, \ldots, m$, by definition)

$$= f(\check{\boldsymbol{\phi}}) - t^r$$

This implies that

$$U_t(\check{\boldsymbol{\phi}}^r, t^r) + t^r \leqslant f(\check{\boldsymbol{\phi}})$$

$$t^{r+1} \leqslant f(\check{\boldsymbol{\phi}})$$

Theorem 2

If t^r is an exact estimate, i.e. $t^r = f(\check{\boldsymbol{\phi}})$ then a solution of $U_t(\boldsymbol{\phi}, t^r)$ is a solution of the non-linear programming problem and vice versa.

Proof

$$U_t(\check{\boldsymbol{\phi}}^r, t^r) \leqslant U_t(\check{\boldsymbol{\phi}}, t^r)$$

$$= 0$$

since $f(\check{\boldsymbol{\phi}}) = t^r$ and $g_i(\check{\boldsymbol{\phi}}) \geqslant 0$, $i = 1, 2, \ldots, m$.

But

$$U_t(\boldsymbol{\phi}, t^r) \geqslant 0$$

Hence

$$U_t(\check{\boldsymbol{\phi}}^r, t^r) = 0$$

This implies that

$$f(\check{\boldsymbol{\phi}}^r) = t^r = f(\check{\boldsymbol{\phi}})$$

and

$$g_i(\check{\boldsymbol{\phi}}^r) \geqslant 0, \quad i = 1, 2, \ldots, m$$

Thus $\check{\boldsymbol{\phi}}^r$ is a solution of the non-linear programming problem.

Conversely,

$$U_t(\check{\boldsymbol{\phi}}, t^r) = 0$$

$$\leqslant U_t(\boldsymbol{\phi}, t^r)$$

Thus, $\check{\boldsymbol{\phi}}$ is a solution of $U_t(\boldsymbol{\phi}, t^r)$.

2.3. *Numerical examples*

Two test functions and a continuous tolerance assignment problem were used to illustrate the performance of the aforementioned algorithms. All the programmes were run on a CDC 6400 computer.

Example 1 : *Beale constrained function* (Kowalik and Osborne 1968)

Minimize

$$f(\boldsymbol{\phi}) = 9 - 8\phi_1 - 6\phi_2 - 4\phi_3 + 2\phi_1^2 + 2\phi_2^2 + \phi_3^2 + 2\phi_1\phi_2 + 2\phi_1\phi_3$$

subject to

$$\phi_i \geqslant 0, \quad i = 1, 2, 3$$

$$3 - \phi_1 - \phi_2 - 2\phi_3 \geqslant 0$$

The function has a minimum $f(\check{\boldsymbol{\phi}}) = \frac{1}{9}$ at $\check{\boldsymbol{\phi}} = [\frac{4}{3}\ \frac{7}{9}\ \frac{4}{9}]^{\mathrm{T}}$. The numerical results from a non-feasible starting point are tabulated in Table 1.

Algorithms	1	2	3	4	5
p value(s)	10^5	10, 10^5	4, 16, 64, 256	10	1·5
Other parameter value(s)	$\alpha=1$	$\alpha=1$	$\alpha=1$ Order of extrapolation = 3	$\alpha=1$	$t^1=0$
ϕ_1	1·3333338	1·3333338	1·3333333	1·3333353	1·3333333
ϕ_2	0·7777775	0·7777772	0·7777778	0·7777776	0·7777778
ϕ_3	0·4444437	0·4444438	0·4444444	0·4444436	0·4444444
$f(\boldsymbol{\phi})$	0·1111114	0·1111114	0·1111111	0·1111111	0·1111111
Number of function evaluations	79	57	45	57	63

Table 1. Comparison of continuous optimization algorithms on Beale function for starting point $\boldsymbol{\phi}^0=[1\ 2\ 1]^T$.

Algorithms	1	2	3	4	5
p value(s)	10^5	10, 10^3, 10^5	4, 16, 64, 256, 1024	10^2	1·5
Other parameter value(s)	$\alpha=10$	$\alpha=10$	$\alpha=10$ Order of extrapolation = 3	$\alpha=10$	$t^1=-50$
ϕ_1	−0·0000021	−0·0000021	−0·0000011	0·0000080	0·0000071
ϕ_2	0·9999976	0·9999976	1·0000035	0·9999996	1·0000033
ϕ_3	1·9999908	1·9999908	1·9999989	2·0000043	2·0000079
ϕ_4	−0·9999883	−0·9999883	−1·0000025	−0·9999653	−0·9999848
$f(\boldsymbol{\phi})$	−43·9998041	−43·9998041	−44·0000025	−43·9999210	−44·0000720
Number of function evaluations	110	90	67	90	300

Table 2. Comparison of continuous optimization algorithms on Rosen–Suzuki function for starting point $\boldsymbol{\phi}^0=[0\ 0\ 0\ 0]^T$.

Algorithm	1	2	3	4	5
p value(s)	10^5	10, 10^3, 10^5	4, 16, 64, 256	4	1·5
Other parameter value(s)	$\alpha=100$	$\alpha=100$	$\alpha=100$ Order of extrapolation = 3	$\alpha=100$	$t^1=0$
ϕ_1	7·58638	7·60603	7·60599	7·60604	7·60600
ϕ_2	9·87321	9·89770	9·89772	9·89771	9·89778
ϕ_3	9·86777	9·89771	9·89772	9·89771	9·89778
ϕ_4	0·90573	0·90564	0·90564	0·90564	0·90563
ϕ_5	2·04267	1·99923	1·99923	1·99923	1·99923
ϕ_6	1·95586	1·99923	1·99923	1·99923	1·99923
$f(\boldsymbol{\phi})$	0·33444	0·33354	0·33354	0·33354	0·33354
Number of function evaluations	700†	425	337	478	157

† 700 is the maximum allowable number of function evaluations.

Table 3. Comparison of continuous optimization algorithms on LC low-pass filter tolerance assignment problem for starting point $\boldsymbol{\phi}^0=[5\ 5\ 5\ 1\ 1\ 1]^T$.

Example 2: *Rosen–Suzuki function* (Kowalik and Osborne 1968)

Minimize

$$f(\boldsymbol{\phi}) = \phi_1^2 + \phi_2^2 + 2\phi_3^2 + \phi_4^2 - 5\phi_1 - 5\phi_2 - 21\phi_3 + 7\phi_4$$

subject to

$$-\phi_1^2 - \phi_2^2 - \phi_3^2 - \phi_4^2 - \phi_1 + \phi_2 - \phi_3 + \phi_4 + 8 \geqslant 0$$

$$-\phi_1^2 - 2\phi_2^2 - \phi_3^2 - 2\phi_4^2 + \phi_1 + \phi_4 + 10 \geqslant 0$$

$$-2\phi_1^2 - \phi_2^2 - \phi_3^2 - 2\phi_1 + \phi_2 + \phi_4 + 5 \geqslant 0$$

The function has a minimum $f(\check{\boldsymbol{\phi}}) = -44$ at $\check{\boldsymbol{\phi}} = [0\ 1\ 2\ -1]^{\mathrm{T}}$. Table 2 shows the performance of the five algorithms from a feasible starting point.

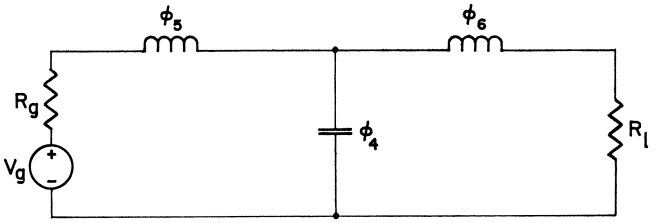

Figure 2. LC low-pass filter used in a tolerance assignment problem.

Example 3: *Tolerance assignment in the design of a low-pass filter* (Bandler 1974, Bandler and Liu 1974)

Consider the low-pass filter shown in Fig. 2. We will minimize the cost function

$$f = \sum_{i=1}^{3} \frac{1}{\phi_i}$$

where ϕ_i is the percentage tolerance of component ϕ_{i+3}.

Let Γ denote the insertion loss. Suppose the pass-band and stop-band specifications are given by

$$\Gamma(\boldsymbol{\phi}, \omega) \leqslant 1{\cdot}5 \text{ dB for } 0 \leqslant \omega \leqslant 1 \text{ rad/sec}$$

and

$$\Gamma(\boldsymbol{\phi}, \omega) \geqslant 25 \text{ dB for } \omega \geqslant 2{\cdot}5 \text{ rad/sec}$$

respectively. A set Ω of five sampling frequency points, namely,

$$\Omega = \{0{\cdot}50,\ 0{\cdot}55,\ 0{\cdot}60,\ 1{\cdot}00,\ 2{\cdot}50\} \text{ rad/sec}$$

was chosen. Minimization was started from a non-feasible point and the results are shown in Table 3.

2.4. *Existence of a feasible solution*

If the constraints cannot be satisfied at the optimum of the least pth objective with any value of p greater than unity, then no feasible solution is attainable (Charalambous and Bandler 1973 a) for all permissible values of p. The existence of a feasible solution may be optionally checked by DISOPT

before solving the non-linear programming problem. DISOPT minimizes with a small value of p the function

$$U_g(\boldsymbol{\phi}) = M_g(\boldsymbol{\phi}) \left[\sum_{j \in J} \left(\frac{-g_j(\boldsymbol{\phi})}{M_g(\boldsymbol{\phi})} \right)^p \right]^{1/p}$$

where

$$M_g(\boldsymbol{\phi}) = \max_{j \in J} [-g_j(\boldsymbol{\phi})]$$

$$J = \{j \,|\, g_j(\boldsymbol{\phi}) \leqslant 0,\ j = 1, 2, \ldots, m\}$$

The minimization terminates if $M_g(\boldsymbol{\phi}) \leqslant 0$ A non-positive value of $M_g(\boldsymbol{\phi})$ at the minimum or even before the minimum is reached indicates that a feasible solution is perceivable. Otherwise, there is no feasible solution to the problem with the current set of constraints.

3. The discrete optimization algorithm

The branch and bound technique was first proposed by Land and Doig (Taha 1971) and later modified by Dakin (1966). The solution of a discrete programming problem by DISOPT follows the logic of this latter approach.

3.1. *Dakin's tree-search algorithm*

The 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. ϕ_i, is non-integral and assumes a value ϕ_i^*, say, in this solution. The range

$$[\phi_i^*] < \phi_i < [\phi_i^*] + 1$$

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

(1) solutions in which
$$\phi_i \leqslant [\phi_i^*]$$
(2) solutions in which
$$\phi_i \geqslant [\phi_i^*] + 1$$

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 non-linear 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 $f(\boldsymbol{\phi})$ 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 discrete variables used. The amount of computation may be vastly different for different trees.

3.2. *Discrete programming*

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

$$\phi_i \geqslant [\phi_i^*/q_i]q_i + q_i$$

and

$$\phi_i \leqslant [\phi_i^*/q_i]q_i$$

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

$$S_i = \{s_1, s_2, \ldots, s_j, s_{j+1}, \ldots, s_d\}, \quad i = 1, 2, \ldots, n$$

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

(1) if $s_j < \phi_i^* < s_{j+1}$, then add the two constraints

$$\phi_i \leqslant s_j$$

and

$$\phi_i \geqslant s_{j+1}$$

sequentially ;

(2) if $\phi_i^* < s_1$, only add the constraint

$$\phi_i \geqslant s_1$$

(3) if $\phi_i^* > s_d$, only add the constraint

$$\phi_i \leqslant s_d$$

The resulting non-linear 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. If an upper bound, $\bar{f}$, on $f(\boldsymbol{\phi})$ is available, then the additional constraint

$$f(\boldsymbol{\phi}) \leqslant \bar{f}$$

is included in the feasibility check. This upper bound, if not specified, will be taken as the current best discrete solution. To obtain an initial upper bound on $f(\boldsymbol{\phi})$ for a discrete problem, DISOPT may be asked to check all the discrete solutions given by letting the variables assume combinations of the nearest upper and lower discrete values (if they exist) and store the best feasible solution.

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. Hence, only a $k-1$ variable problem need be solved and much computational effort would be saved.

3.3. *Numerical examples*

Four discrete minimization problems have been included here to demonstrate the use of the programme.

Example 1 : *Modified banana shape function*

Minimize

$$f(\boldsymbol{\phi}) = 100((\phi_2 + 0{\cdot}5) - (\phi_1 + 0{\cdot}6)^2)^2 + (0{\cdot}4 - \phi_1)^2$$

subject to

$$\phi_1,\ \phi_2 \text{ natural numbers}$$

The results are tabulated in Table 4. This example serves to illustrate that the optimum discrete solution is not guaranteed by simply chopping or rounding off the continuous solution. From the contour plot shown in Fig. 3

Solution	Continuous	Discrete
ϕ_1	0·4000	1
ϕ_2	0·5000	2
$f(\boldsymbol{\phi})$	0·0000	0·72
Function evaluations		878
Nodes		9
Time (sec)		8

Table 4. Results for example 1 starting at $\boldsymbol{\phi}^0 = [-1{\cdot}8\ \ 0{\cdot}5]^T$ and using algorithm 3. $p^1 = 4,\ c = 4$.

it is obvious that the optimum discrete solution is not given by any of the vertices about the continuous solution. The best vertex is given by $\boldsymbol{\phi} = [0\ 0]^T$ with a function value $f(\boldsymbol{\phi}) = 2{\cdot}12$ which is much higher than that for the optimum discrete solution.

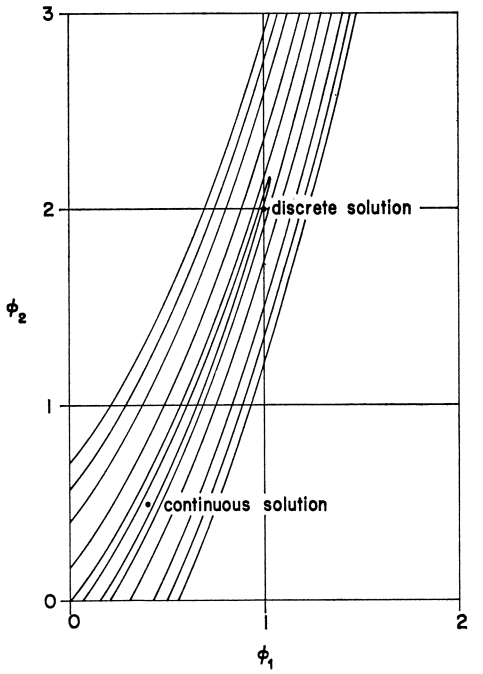

Figure 3. Contour plot for the modified banana shape function.

Example 2: Beale constrained function

Minimize the Beale function subject to the additional constraint that the variables must be integers. The results are shown in Table 5. All the three optimum discrete solutions of unity function value are detected by the algorithm. However, if the user indicates that only one optimum discrete solution is required, DISOPT will check the existence of a better solution before solving the non-linear programming problem at a node. As illustrated by this example, this will reduce the necessary computational effort.

Solution	Continuous	Discrete		
ϕ_1	1·3333	2	1	2
ϕ_2	0·7778	0	1	1
ϕ_3	0·4444	0	0	0
$f(\boldsymbol{\phi})$	0·1111	1	1	1
Number of optimum discrete solutions required			3	1
Function evaluations			226	160
Nodes			7	7
Time (sec)			5	4

Table 5. Results for example 2 starting at $\boldsymbol{\phi}^0=[1\ 2\ 1]^T$ and using algorithm 1. $p=10^3$.

Example 3: *Tolerance assignment in the design of a voltage divider* (Karafin 1972)

Consider the simple voltage divider as shown in Fig. 4. The transfer function is given by $T=\phi_4/(\phi_3+\phi_4)$ and the input resistance is $R=\phi_3+\phi_4$. The design specifications are $0{\cdot}46 \leqslant T \leqslant 0{\cdot}53$ and $1{\cdot}85 \leqslant R \leqslant 2{\cdot}15$. The obtainable discrete tolerances for both ϕ_3 and ϕ_4 are given by the set

$$S=\{1, 3, 5, 10, 15\} \text{ per cent}$$

The cost function

$$f=\sum_{i=1}^{2}\frac{1}{\phi_i}$$

where ϕ_i is the percentage tolerance in component ϕ_{i+2}, was first minimized by fixing one variable at each node in the search for discrete solution. The minimization was then repeated as a four-dimensional problem throughout to highlight the extra amount of effort that was required. The numerical results are shown in Table 6.

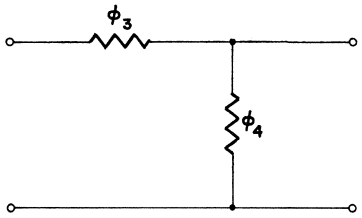

Figure 4. Voltage divider used in a tolerance assignment problem.

Solution	Continuous	Discrete	
ϕ_1	7·0007	5	
ϕ_2	7·0007	5	
ϕ_3	1·0137		
ϕ_4	0·9935		
f	0·2854	0·4	
Dimensionality of the problem used in the search for discrete solution		3	4
Function evaluations		589	1083
Nodes		9	9
Time (sec)		10	17

Table 6. Results for example 3 starting at $\boldsymbol{\phi}^0=[1\ 1\ 1\ 1]^T$ and using algorithm 4. $p=6$.

Example 4: *Tolerance assignment in the design of a low-pass filter*

The cost function for the previous tolerance assignment problem in filter design was minimized with the additional constraint that only the following set, S, of discrete tolerances was available for each of the components:

$$S=\{1, 2, 5, 10, 15\} \text{ per cent}$$

The numerical results are tabulated in Table 7. This example illustrates that the tree structure and hence the computational effort is dependent upon the order of partitioning on the discrete variables.

Solution	Continuous	Discrete		
ϕ_1	7·6061	5	10	10
ϕ_2	9·8978	10	5	10
ϕ_3	9·8978	10	10	5
ϕ_4		0·9056		
ϕ_5		1·9992		
ϕ_6		1·9992		
f	0·3335		0·4	
The discrete variable first used for constructing the variable constraints			ϕ_1	ϕ_3
Function evaluations			3704	3314
Nodes			27	23
Time (sec)			91	82

Table 7. Results for example 4 starting at $\boldsymbol{\phi}^0=[5\ 5\ 5\ 1\ 1\ 1]^T$ and using algorithm 5. $p=1{\cdot}5$.

4. Conclusions

An integrated optimization programme called DISOPT is presented in this paper. Many up to date algorithms and techniques have been incorporated into one programme and made available to the user. Illustrative examples have been included to demonstrate the efficiency of DISOPT and the various options present.

An unfortunate characteristic of optimization is that no one technique is best for all kinds of problems. Hence, it is advantageous to have a multi-technique general programme. From the authors' experience, algorithm 5 should be recommended only if a good optimistic estimate of the optimum function value is available. Otherwise, the minimax approach to non-linear programming should be used. If the starting point is not likely to lie in the close vicinity of the optimum, a sequence of least pth optimizations should be used to avoid poor scaling of the problem. However, if the starting point happens to be very close to the optimum, the use of small values of p in the initial optimizations will actually give worse estimates of the optimum.

The amount of programming effort required of the user has been reduced to a minimum. A user is responsible only for

(1) supplying the values and/or proper dimensioning of the parameters in the argument list ; and
(2) writing two service subroutines to define the objective function, the constraints and their respective partial derivatives.

DISOPT will, on exit, output the required solution or a message if a solution does not exist. Since all the input data is entered through the argument of DISOPT, the programme can be easily incorporated into other user-oriented computer-aided design packages. The complete listing and documentation are available (Chen 1974).

References

Bandler, J. W., 1974, *J. Optimiz. Theory Applic.*, **14,** 99.

Bandler, J. W., and Charalambous, C., 1972, *I.E.E.E. Trans. microw. Theory Tech.*, **20,** 834 ; 1974, *J. Optim. Theory Applic.*, **13,** 607.

Bandler, J. W., and Chu, W. Y., 1974, *Proc. Twelfth Allerton Conf. on Circuit and System Theory*, Urbana, Illinois.

Bandler, J. W., and Liu, P. C., 1974, *I.E.E.E. Trans. Circuits Systems*, **21,** 219.

Charalambous, C., 1974 a, *I.E.E.E. Trans. microw. Theory Tech.*, **22,** 289 ; 1974 b, Department of Combinatorics and Optimization, Research Report 74–2 (Waterloo, Canada : University of Waterloo).

Charalambous, C., and Bandler, J. W., 1972, Internal Report (Hamilton, Canada : Department of Electrical Engineering, McMaster University) ; 1973 a, *I.E.E.E. Trans. microw. Theory Tech.*, **21,** 815 ; 1973 b, Internal Report in Simulation, Optimization and Control, SOC–3 (Hamilton, Canada : McMaster University).

Chen, J. H. K., 1974, Internal Report in Simulation, Optimization and Control, SOC–29 (Hamilton, Canada : McMaster University).

Dakin, R. J., 1966, *Computer J.*, **8,** 250.

Fletcher, R., 1970, *Computer J.*, **13,** 317 ; 1972, Report AERE–R7125 (Harwell, Berkshire : Atomic Energy Research Establishment).

Fiacco, A. V., and McCormick, G. P., 1968, *Nonlinear Programming : Sequential Unconstrained Minimization Techniques* (New York : Wiley).

Gill, P. E., and Murray, W., 1972, *J. Inst. Maths. Applic.*, **9,** 91.

Karafin, B. J., 1972, *B.S.T.J.*, **50,** 1225.

Kowalik, J., and Osborne, M. R., 1968, *Methods for Unconstrained Optimization Problems* (New York : Elsevier).

Lootsma, F. A., 1972, *Numerical Methods for Nonlinear Optimization*, edited by F. A. Lootsma (New York : Academic Press).

Taha, H. A., 1971, *Operations Research—An Introduction* (New York : MacMillan).