SIAM J. SCI. STAT. COMPUT.
Vol. 6, No. 1, January 1985

THE TUNNELING ALGORITHM FOR THE GLOBAL MINIMIZATION OF FUNCTIONS*

A. V. LEVY† AND A. MONTALVO‡

Abstract. This paper considers the problem of finding the global minima of a function $f(x): \Omega \subset \mathcal{R}^n \to \mathcal{R}$. For this purpose we present an algorithm composed of a sequence of cycles, each cycle consisting of two phases: (a) a minimization phase having the purpose of lowering the current function value until a local minimizer is found and, (b) a tunneling phase that has the purpose of finding a point $x \in \Omega$, other than the last minimizer found, such that when employed as starting point for the next minimization phase, the new stationary point will have a function value no greater than the previous minimum found.

In order to test the algorithm, several numerical examples are presented. The functions considered are such that the number of relative minima varies between a few and several thousand; in all cases, the algorithm presented here was able to find the global minimizer(s). When compared with alternate procedures, the results show that the new algorithm converges more often to the global minimizer(s) than its competitors; additionally, it becomes more efficient than the other procedures for problems with increasing density of relative minima.

Key words. global optimization, nonlinear programming, unconstrained minimization

1. Introduction. One of the most interesting research areas in nonlinear programming is that of finding the global minimizer of a function defined in an n-dimensional linear space. Until recent years this problem has been almost completely ignored and only few numerical methods have been developed to solve it. Among these, three approaches have been used: (a) In the hill climbing approach, all the extremal points are found sequentially in a deterministic fashion and comparing the values of $f(x)$ at *all* the local minimizers, the global is identified (Brannin [3]; Hardy [6]). (b) In the multiple random start approach, a very large number of starting points are given and any minimization algorithm is used to find the corresponding local minimizer. By comparing the values of $f(x)$ at *all* the local minimizers the global is identified (Anderssen [1]). (c) In the function modification approach, an auxiliary function is defined as whose function value at the local minimizer is lower than the value of the function at all the local minimizers previously found. Using this algorithm recursively, the global could be approached (Goldstein [5]).

Most of the algorithms so far developed along these lines, are practical only for low dimensionality problems. In this paper we present a new algorithm which retains the best properties of each approach, while avoiding their main disadvantages, such as unpredictable performance while approaching the global minimizer(s), large computing time and the evaluation of higher order derivatives. The new method (Tunneling Algorithm) consists of two phases, a minimization phase and a tunneling phase. These phases are used sequentially to approach the global minimizer of $f(x)$. In the minimization phase, for a given starting point x^0, any minimization algorithm with a local descent property on $f(x)$ can be used to find a local minimum of $f(x)$, say at x^*.

In the tunneling phase, an auxiliary function $T(x)$, is defined, where T, the tunneling function, is a scalar function with continuous first derivatives, whose zero-set

* Received by the editors March 15, 1982, and in revised form August 23, 1983. This work was partially supported by the National Science Foundation Grant NSF-MCS-76-21657.

† Postdoctoral Fellow in Aero-Astronautics, Rice University, Houston, Texas, Professor of IIMAS, Universidad Nacional Autónoma de Mexico, Mexico City, DF, Mexico.

‡ Research Associate, IIMAS, Universidad Nacional Autónoma de Mexico, Apdo. Postal 20-726, Deleg. A. Obregón, 01000 Mexico, DF, Mexico.

coincides with the set where $f(x)=f(x^*)$ and which depends on a set of parameters that are chosen automatically by the algorithm with the purpose of stabilizing the method. The objective of this phase is the following: starting at any point in a neighborhood of x^*, we seek a new point x^0 such that $T(x^0)\leqq 0$.

By using these two phases alternately, the sequence of minimizers found are such that the function value at each of these minimizers is never greater than any of the function values at the previously found minimizers; that is, the function value at the local minimizers is nonincreasing.

Numerical results are presented for sixteen examples which are solved by the present method and by two versions of the multiple random start approach. These examples vary from two to ten variables, from one strict global minimum to eighteen global minima and up to several thousands of stationary points.

The numerical results show that the present method is usually faster than the other methods and more important, it converges more often to the global minimizer than the other methods. In particular, the numerical results indicate that these relative advantages increase with the density of relative minima.

2. Derivation of the algorithm. Let $f(x)$ be a twice continuously differentiable function on the set $\Omega=\{x\in\mathbb{R}^n: a\leqq x\leqq b\}$, with a and $b\in\mathbb{R}^n$. We will assume that all the minima of $f(x)$ are isolated minima and that there is a finite number of them.

In this paper we consider the problem of finding all the global minimizers of $f(x)$ in Ω. For solving this problem, we will present an algorithm that consists of two separate phases. The objective of the first phase is to find a local minimizer x^* of $f(x)$; the second phase is designed to move from this point to a point $x^0\neq x^*$ with $f(x^0)\leqq f(x^*)$. These two phases are described next.

2.1. Minimization phase. Given a starting point x^0 we use any minimization algorithm to find a local minimizer of $f(x)$, say x^*. This algorithm can be any unconstrained minimization method with descent property, e.g., steepest descent, conjugate gradient, Newton's method. We will assume that at the end of this phase a local minimizer has been found.

2.2. Tunneling phase. We start this phase at x^*, the exit point of the minimization phase, and its purpose is to find a point $x^0\in\Omega$ such that

$$\begin{aligned} &f(x^0)\leqq f(x^*),\\ &x^0\neq x^*. \end{aligned} \tag{1}$$

This can be formally stated as follows: find $x^0\in Z=\{x\in\Omega-\{x^*\}: f(x)\leqq f(x^*)\}$. We now have the following dichotomy, whenever Z is not empty:

i) x^0 is also a local minimizer of $f(x)$ and $x^0\neq x^*$;

ii) if x^0 is not a minimizer of $f(x)$; then a further descent on the function value can be achieved, e.g., by moving along the direction given by $-f_x(x^0)$.

In the former case we have located a new minimizer of $f(x)$ where the function value *is not higher* than the previous one, $f(x^*)$; in the latter case we have found a point that can be used to start a new minimization phase that will locate a new minimizer with a function value lower than $f(x^*)$. (In the case when $Z=\varnothing$, then the algorithm goes to the boundary of Ω.)

In order to move from x^*, consider the function

$$T(x)=\frac{f(x)-f(x^*)}{[(x-x^*)'(x-x^*)]^{\eta}} \tag{2}$$

(where $'$ denotes transposition) which has a pole at x^* for η sufficiently large. Note that all $x^0 \neq x^*$ satisfying $T(x^0) \leqq 0$ are contained in Z. Therefore, we will consider the problem of finding a nonpositive minimum of $T(x)$. In practice, after several applications of the two phases the function $T(x)$ will have the general form (Appendix I)

$$T(x) = \frac{f(x) - f^*}{\{\prod_{i=1}^{l} [(x - x_i^*)'(x - x_i^*)]^{\eta_i}\}[(x - x_m)'(x - x_m)]^{\lambda_0}}. \tag{3}$$

The purpose of each of the terms in (3) is the following: The difference in the numerator eliminates as possible solutions of this phase all those points x satisfying $f(x) > f^*$. The first term in the denominator prevents the algorithm to locate as solutions of this phase all previous minimizers found at x_i^*, $i = 1, 2, \cdots, l$, with a function value $f(x_1^*) = f(x_2^*) = \cdots = f(x_l^*) = f^*$; that is, the algorithm will not cycle between previous solutions with the same function value (*Remark.* l is always taken as one whenever the new minimizer found produces a function value strictly lower than the previous one, and is increased by one if the new minimizer produces a function value equal to the previous one). The second term in the denominator of (3) is designed to smooth out, in an adaptive fashion, any irrelevant local minimizer of $T(x)$ that might attract any particular minimization algorithm during the search for x^0 (points where $T_x(x) = 0$ and $T(x) > 0$).

Finally, the tunneling phase can be actually implemented by using any minimization algorithm with descent property on $T(x)$. The rules needed to determine the correct value of all the parameters involved in $T(x)$ are presented in Appendix I.

2.3. Minimization algorithms. Next, we present three different algorithms employed in the minimization phase; namely, ordinary gradient, conjugate gradient and Newton's methods as well as a modified version of Newton's method as employed in the tunneling phase.

2.3.1. Ordinary gradient (Himmelblau [7]). For this method, the displacements Δx are generated according to

$$\Delta x = -\alpha f_x(x)$$

where α, the step size, is chosen such that

$$f(x + \Delta x) < f(x)$$

by using a bisection procedure on α, starting at $\alpha = 1$ and up to N_b bisections.

2.3.2. Conjugate gradient. This method was implemented following the Fletcher–Reeves method (Himmelblau [7]) as follows:

a. Set $k = 0$ and $\Delta x^0 = -f_x(x^0)$ as the first search direction;

b. find α, the step size, such that

$$\frac{df(x^k + \alpha \Delta x^k)}{d\alpha} = 0;$$

c. $x^{k+1} = x^k + \alpha \Delta x^k$;

d. The new search direction is evaluated according to

$$\Delta x^{k+1} = -f_x(x^{k+1}) + \frac{f_x'(x^{k+1}) f_x(x^{k+1})}{f_x'(x^k) f_x(x^k)} \Delta x^k;$$

e. $k = k + 1$ and repeat the procedure from step b. Whenever $k = n + 1$, the entire procedure is restarted at step a, by taking $x^0 = x^k$.

Finally, to accomplish step b, the minimum of $f(x)$ along the search direction Δx^k, is first bracketed; once this is done, the optimum value of α is found by using quadratic interpolation.

2.3.3. Newton's method (Himmelblau [7]). This method generates displacements Δx according to

$$\Delta x = -\alpha\rho f_{xx}^{-1}(x) f_x(x)$$

where α, the step size is determined as in the ordinary gradient method and ρ, a direction factor, is defined according to

$$\rho = \begin{cases} 1 & \text{if } f(x - 2^{-N_b} f_{xx}^{-1}(x) f_x(x)) < f(x), \\ -1 & \text{if } f(x - 2^{-N_b} f_{xx}^{-1}(x) f_x(x)) > f(x). \end{cases}$$

2.3.4. Modified Newton's method (Miele [8]). The displacements Δx, in the tunneling phase are generated according to

$$\Delta x = -\alpha \frac{T(x)}{T'_x(x)\, T_x(x)} T_x(x)$$

and α, the step size, is determined as in the ordinary gradient and Newton's methods.

In all four methods described above, provisions must be taken in order to prevent that any point generated by the procedures lies outside the admissible region.

3. Some properties of the tunneling algorithm. Using the material presented in the previous section, we now exhibit the following descent property of the tunneling algorithm obtained by successive applications of minimization and tunneling phases.

a) If any of the minimization phases uses x_i^0 as a starting point, a local minimizer of $f(x)$ will be found, say x_i^*, whose function values satisfy

$$f(x_i^*) \leqq f(x_i^0). \tag{4}$$

b) Similarly, if any of the tunneling phases uses x_i^* as starting point, the solution point of the corresponding tunneling function $T(x)$, say x_{i+1}^0, satisfies

$$f(x_{i+1}^0) \leqq f(x_i^*), \qquad x_{i+1}^0 \neq x_i^*. \tag{5}$$

c) Combining (4) and (5), and suppressing the intermediate solution of the tunneling phases, the descent property of the algorithm is given by

$$f(x_1^*) \geqq f(x_2^*) \geqq \cdots \geqq f(x_{G-1}^*) \geqq f(x_G^*) \tag{6}$$

where x_G^* corresponds to the global minimizer of $f(x)$ and where $x_i^* \neq x_j^*$ for $i \neq j$.

4. Numerical experiments. It is the purpose of this section to illustrate the characteristics of the Tunneling Algorithm as well as the methods already mentioned in § 1, namely, two versions of the random starting method. For this purpose, sixteen numerical examples were solved in a CDC-6400 computer, using single precision arithmetic, with all the algorithm coded in standard FORTRAN IV.

4.1. Stopping conditions.

4.1.1. Tunneling algorithm. The value assigned to the different parameters involved in the minimization phase were the following: the convergence criteria were chosen as

$$\varepsilon_1 = 10^{-9}$$

and the nonconvergence criteria were set to

$$N_b = 20.$$

The values assigned to those parameters involved in the tunneling phase are given in Appendix I.

4.1.2. Multiple random start method (MRS). This algorithm is made-up of a random number generator which gives the starting point of any minimization phase. The actual implementation of this technique is the following. For a given minimization algorithm we ran it for different starting points, selected at random, and stored the obtained local minima of $f(x)$ and their location. (Note that in this method neither the number of starting points nor their location are known in advance.)

The stopping condition employed in this algorithm to denote convergence at any local minimum was

$$f'_x(x) f_x(x) \leqq \varepsilon_1 = 10^{-9}. \tag{7}$$

Nonconvergence is defined to occur whenever the number of bisections in a given minimization step is greater than 20 or when the limiting computing time is reached.

4.1.3. Modified multiple random start method (MMRS). This algorithm is in the same line as the previous one. The only modification involved in MMRS is as follows: Let x^* be the location where a minimum of $f(x)$ occurs, say $f(x^*)$. To continue the search for additional minima, a random point x^0 is generated and used as the location to start the next minimization phase provided

$$f(x^0) - f(x^*) \leqq 10^{-3} \tag{8}$$

is satisfied. If for a given x^0, (8) is not satisfied, a new point x^0 is generated at random and satisfaction of (8) is tested; this procedure is repeated until inequality (8) is satisfied or a time limit is reached. (As in the MRS method neither the number of starting points nor their location are known in advance.)

4.2. Measurement of success. Each of the problems presented in § 4.4 was solved several times starting each run at different points. Let N_G denote the number of global minima exhibited by a particular problem ($N_G = 1$ for problems with one strict global minimum); N_r the number of starting points employed in each problem and M_i, $i = 1, 2, \cdots, N_r$ the number of different global minima found when starting at each one of the N_r starting points; under these conditions we define a quantity p, a measure of success, as

$$p = \frac{\sum_{i=1}^{N_r} M_i}{N_G N_r}. \tag{9}$$

From this equation we observe that $0 \leqq p \leqq 1$, and the larger value of p, the more robust is the algorithm since more often finds all the N_G global minimizers.

4.3. Running time. Each of the problems presented in the following section was first solved using the tunneling algorithm. Let t_i, $i = 1, 2, \cdots, N_r$ denote the time consumed by the tunneling algorithm, in CPU secs, from the beginning of the first minimization phase up to the end of the last minimization phase. With these times, we define an average running time as

$$t_{\text{av}} = \frac{\sum_{i=1}^{N_r} t_i}{N_r}. \tag{10}$$

This average time, t_{av}, was employed as limiting time when running the alternate methods, giving this a basis to compare all different methods.

4.4. Examples. In this section we describe the sixteen examples employed to compare the algorithms. The region of interest, where the examples were considered as well as the starting points employed in each example, is given.

Example 1. "*Two-dimensional Shubert function*" (Shubert [10]).

$$f(x_1, x_2) = \left\{ \sum_{i=1}^{5} i \cos[(i+1)x_1 + i] \right\} \left\{ \sum_{i=1}^{5} i \cos[(i+1)x_2 + i] \right\}, \tag{11}$$

$$-10 \leqq x_i \leqq 10, \quad i = 1, 2.$$

This function exhibits 760 local minima in the region considered, eighteen of which are also global minima. The value of $f(x)$ at these global minima is $f(x_1, x_2) = -186.73091$ and the coordinates of each one of the minimizers are the following:

$(-7.08350, -7.70831)$, $(-0.80032, -7.70831)$, $(5.48286, -7.70831)$,

$(-7.70831, -7.08350)$, $(-1.42513, -7.08350)$, $(4.85805, -7.08350)$,

$(-7.08350, -1.42513)$, $(-0.80032, -1.42513)$, $(5.48286, -1.42513)$,

$(-7.70831, -0.80032)$, $(-1.42513, -0.80032)$, $(4.85805, -0.80032)$,

$(-7.08350, 4.85805)$, $(-0.80032, 4.85805)$, $(5.48286, 4.85805)$,

$(-7.70831, 5.48286)$, $(-1.42513, 5.48286)$, $(4.85805, 5.48286)$.

The starting points for this example were $(7, 7)$, $(7, -7)$, $(-7, 7)$, $(0, 0)$.

Example 2. "*Two-dimensional Shubert function*".

Case 1.

$$f(x_1, x_2) = \left\{ \sum_{i=1}^{5} i \cos[(i+1)x_1 + i] \right\} \left\{ \sum_{i=1}^{5} i \cos[(i+1)x_2 + i] \right\} \tag{12}$$

$$+ \tfrac{1}{2}[(x_1 + 1.42513)^2 + (x_2 + 0.80032)^2], \qquad -10 \leqq x_i \leqq 10, \quad i = 1, 2.$$

This function exhibits the same characteristics as Example 3 with only one global minimum located at

$$x_1 = -1.42513, \qquad x_2 = -0.80032$$

with a function value $f(x) = -186.73091$. The starting points were the same as those in Example 1.

Example 3. "*Two-dimensional Shubert function*".

Case 2.

$$f(x_1, x_2) = \left\{ \sum_{i=1}^{5} i \cos[(i+1)x_1 + i] \right\} \left\{ \sum_{i=1}^{5} i \cos[(i+1)x_2 + i] \right\} \tag{13}$$

$$+ (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2, \qquad -10 \leqq x_i \leqq 10, \quad i = 1, 2.$$

This example has the same characteristics of Examples 3 and 4, with only one global minimum located at

$$x_1 = -1.42513, \qquad x_2 = -0.80032$$

with a function value $f(x) = -186.73091$. The starting points were the same as for Example 1.

Example 4. "*Six-hump camelback function*" (Hardy [6]).

$$f(x_1, x_2) = [4 - 2.1x_1^2 + \tfrac{1}{3}x_1^4]x_1^2 + x_1x_2 + [-4 + 4x_2^2]x_2^2, \quad -3 \leqq x_1 \leqq 3, \quad -2 \leqq x_2 \leqq 2. \tag{14}$$

This function exhibits 6 local minimizers, two of which are also global, located at $(-0.08983, 0.7126)$ and $(0.08983, -0.7126)$ and the function value is $f(x) = -1.0316285$.

The starting points employed in this example were $(-2.9, -1.9)$, $(-2.9, 1.9)$, $(2.9, -1.9)$, and $(2.9, 1.9)$.

Examples 5 through 7 are taken from the general formula

$$f(x) = \frac{\pi}{n}\left\{ k \sin^2(\pi y_1) + \sum_{i=1}^{n-1} [(y_i - A)^2(1 + k \sin^2(\pi y_{i+1}))] + (y_n - A)^2 \right\},$$
$$y_i = 1 + 0.25(x_i - 1), \quad -10 \leqq x_i \leqq 10, \quad i = 1, 2, \cdots, n \tag{15}$$

where the constants k and A were fixed at 10 and 1 respectively, and n denotes the dimensionality of the problem.

The function given by (15) exhibits many local minima but only one of these is also a global minimum. The location of this global minimizer is fixed at

$$x_i = 1, \qquad i = 1, 2, \cdots, n$$

and the function attains the value $f(x) = 0$, irrespective of the dimensionality of the problem.

Example 5. *Equation* (15), $n = 2$. The starting points used in this example were $(-8, 8)$, $(8, 8)$, $(-5, 5)$, and $(-8, 8)$.

Example 6. *Equation* (15), $n = 3$. The starting points used in this example were $(8, 8, 8)$, $(-5, 5, -5)$, $(8, -8, 8)$, and $(-8, -8, -8)$.

Example 7. *Equation* (15), $n = 4$. The starting points used in this example were $(-5, -5, -5, -5)$, $(5, 5, 5, 5)$, $(-5, -5, 5, 5)$, and $(5, 5, -5, -5)$.

Examples 8 through 10 are taken from the general formula

$$f(x) = \frac{\pi}{n}\left\{ k \sin^2(\pi x_1) + \sum_{i=1}^{n-1} [(x_i - A)^2(1 + k \sin^2(\pi x_{i+1}))] + (x_n - A)^2 \right\},$$
$$-10 \leqq x_i \leqq 10, \quad i = 1, 2, \cdots, n, \tag{16}$$

where the constants k and A were fixed at 10 and 1 respectively and n denotes the dimensionality of the problem.

Example 8. *Equation* (16), $n = 5$. The starting points used for this example were $(8, 8, 8, 8, 8)$, $(-8, -8, 0, 8, 8)$, $(8, 8, 0, -8, -8)$, and $(-8, -8, -8, -8, -8)$.

Example 9. *Equation* (16), $n = 8$. The starting points used in this example were

a) $x_i = 8, \quad i = 1, 2, \cdots, 8,$

b) $x_i = -8, \quad i = 1, 2, \cdots, 6, \quad x_i = 0, \quad i = 7, 8,$

c) $x_i = 8(-1)^i, \quad i = 1, 2, \cdots, 6, \quad x_i = 0, \quad i = 7, 8,$

d) $x_i = 0, \quad i = 1, 2, \cdots, 8.$

Example 10. *Equation* (16), $n = 10$. The starting points used in this example were

a) $x_i = 0, \quad i = 1, 2, \cdots, 10,$

b) $x_i = 2, \quad i = 1, 2, \cdots, 10,$

c) $x_i = 6, \quad i = 1, 2, \cdots, 10,$

d) $x_i = -1, \quad i = 1, 2, \cdots, 10.$

Examples 11 through 16 have been taken from the general formula

$$f(x) = k_1 \sin^2 \pi l_0 x_1 + k_1 \sum_{i=1}^{n-1} [(x_i - A)^2 (1 + k_0 \sin^2 \pi l_0 x_{i+1})]$$
$$+ k_1 (x_n - A)^2 (1 + k_0 \sin^2 \pi l_0 x_n), \tag{17}$$

where the constants in this equation have been fixed as follows: $k_0 = 1$, $k_1 = 0.1$, $A = 1$, $l_0 = 3$, and $l_1 = 2$. The examples generated by using (17) exhibit many local minimizers, but only one of these is also global located at $x_i^* = 1$, $i = 1, 2, \cdots, n$ and the function value at this point is $f(x^*) = 0$.

Example 11. *Equation* (17), $n = 2$, $-10 \leqq x_i \leqq 10$, $i = 1, 2$. The starting points for this example were $(9, 9)$, $(-9, -9)$, $(-9, 9)$, and $(9, -9)$.

Example 12. *Equation* (17), $n = 3$, $-10 \leqq x_i \leqq 10$, $i = 1, 2, 3$. The starting points were $(5, 5, 5)$, $(5, -5, 5)$, $(-5, 5, -5)$, and $(-5, -5, -5)$.

Example 13. *Equation* (17), $n = 4$, $-10 \leqq x_i \leqq 10$, $i = 1, 2, 3, 4$. The starting nominal points for this example were $(5, 5, 5, 5)$, $(5, 5, -5, -5)$, $(-5, -5, 5, 5)$, and $(-5, 0, 0, 5)$.

Example 14. *Equation* (17), $n = 5$, $-5 \leqq x_i \leqq 5$, $i = 1, 2, \cdots, 5$. The starting points for this example were

a) $x_i = 3$, $\quad i = 1, 2, \cdots, 5$,
b) $x_i = 3(-1)^{i-1}$, $\quad i = 1, 2, \cdots, 5$,
c) $x_i = 3(-1)^i$, $\quad i = 1, 2, \cdots, 5$,
d) $x_i = -3$, $\quad i = 1, 2, \cdots, 5$.

Example 15. *Equation* (17), $n = 6$, $-5 \leqq x_i \leqq 5$, $i = 1, 2, \cdots, 6$. The starting points used for this example were

a) $x_i = 3$, $\quad i = 1, 2, \cdots, 6$,
b) $x_i = 3$, $\quad i = 1, 2, 3$, $\quad x_i = -3$, $\quad i = 4, 5, 6$,
c) $x_i = -3$, $\quad i = 1, 2, 3$, $\quad x_i = 3$, $\quad i = 4, 5, 6$,
d) $x_i = -3$, $\quad i = 1, 2, \cdots, 6$.

Example 16. *Equation* (17), $n = 7$, $-5 \leqq x_i \leqq 5$, $i = 1, 2, \cdots, 7$. The starting points for this example were

a) $x_i = -3$, $\quad i = 1, 2, \cdots, 7$,
b) $x_i = -3$, $\quad i = 1, 2, 3$, $\quad x_4 = 0$, $\quad x_i = 3$, $\quad i = 5, 6, 7$,
c) $x_i = 3$, $\quad i = 1, 2, 3$, $\quad x_4 = 0$, $\quad x_i = -3$, $\quad i = 5, 6, 7$,
d) $x_i = 3$, $\quad i = 1, 2, \cdots, 7$.

5. Analysis of the numerical results. In this section we present the results obtained for the sixteen examples mentioned in the previous section when solved using the various methods previously described. Additionally we present the result obtained when running the tunneling algorithm coupled to several local minimization techniques.

Next, we present the results obtained by the tunneling, MRS and MMRS methods, using in their corresponding minimization phases the ordinary gradient method to find the local minimizers (Montalvo [9]).

With the purpose of having as much information as possible to have a better idea on the performance of the methods, we report the following data for each one of the sixteen examples.

a) Total time required for reaching the global minimizers.
b) Total number of functions and gradient evaluations.
c) Partial time spent in the minimization phases.
d) Total number of functions and gradient evaluation during the minimization phases.

e) Minimization phases required to reach the global minimizers.

f) Measurement of success (p). Here we have to differentiate the following situations:

f.1) For functions exhibiting only one global minimum, the reported value for the tunneling algorithm varies between zero and one, since this value corresponds to the number of times that, in the average, the global minimum was reached starting at four different points. For MRS and MMRS methods, the reported values are either zero or one depending whether or not the global minimum was located or missed.

f.2) For functions exhibiting more than one global minimum the reported value is obtained as follows: for each one of the runs starting at different points, we counted the number of different global minima found and then an average between these four numbers is computed; the final number is obtained dividing the latter by the total number of global minima exhibited by the function.

From the results shown in Table 1, we can make the following observations.

a) In all sixteen examples, the tunneling algorithm was the only one in locating all global minimizers, except in Example 1 where the method located seventeen out of the eighteen globals.

b) On Examples 5, 6, 7, 11 and, 12, the MMRS algorithm out-performed the tunneling algorithm as it was able to find the global minimizers faster than the tunneling algorithm.

c) On Examples 2, 3, 8, 10, and 13–16 both MRS and MMRS were not able to locate the global minimizers, while the tunneling algorithm did; these examples include the one with $n = 10$.

Taking into account the above statements we reach the conclusion that the tunneling algorithm is usually faster and more often converges to the global minimizers than the other methods; this advantage increases with the density of relative minima. In particular, the starting point generated by the tunneling algorithm (exit point generated by the tunneling phase), can be considered as a "highly educated" random starting point for initiating each of the minimization phases.

Use of alternate local minimization techniques. In Table 2 we present the results obtained when running the tunneling algorithm coupled to different local minimization techniques, namely: (a) ordinary gradient, (b) conjugate gradient and (c) Newton's method.

For the ten examples analyzed (Examples 1–10) we have focused the comparison on the time spent by the algorithms in (a) the minimization phases and (b) reaching the global; that is, we allowed the program to reach the global minimizer. For each case we report the above mentioned two times as well as the average number of minimizations performed and the average number of times the global minimizer was located.

We observe that in all cases the tunneling algorithm was always able to locate the global, independently of the local minimization technique being used. However, one can observe the following anomalous results in Table 2.

a) The ordinary gradient method took the least total times for reaching the global in Examples 1, 3, 8, and 10.

b) The conjugate gradient method was the best in Examples 2, 3, and 9.

c) Newton's method became the best method for Examples 4, 5, 6, and 7.

Another important fact that can also be observed in Table 2, is that in seven of the ten examples, the ordinary gradient method required the least minimization phases. It is the authors' opinion that this characteristic was probably due to the fact that even during the search for local minimizers the minimization method was also "tunneling";

TABLE 1
Summary of numerical results for the tunneling, MRS, and MMRS methods.

Ex/dim.	Method	Total time (CPU-secs)	Total evaluations		Minimization time (CPU-secs)	Evaluations during minimization phase		Number of minimizations	p
			Functions	Gradient		Functions	Gradient		
1/2	Tunnel	87.045	12,160	1,731	2.113	1,047	97	17	0.9445
	MRS	88.089	35,479	9,572	87.845	35,479	9,572	244	0.5
	MMRS	87.066	50,462	7	0.148	75	7	1	0.0555
2/2	Tunnel	8.478	2,912	390	1.045	525	53	3	1
	MRS	5.197	*	*	*	*	*	10	0
	MMRS	4.313	*	*	*	*	*	2	0
3/2	Tunnel	5.984	2,180	274	1.409	710	69	3	1
	MRS	2.094	*	*	*	*	*	4	0
	MMRS	6.018	3,350	19	0.292	136	19	1	0
4/2	Tunnel	1.984	1,496	148	0.033	57	17	2	1
	MRS	0.036	61	19	0.034	61	19	2	1
	MMRS	2.036	6,000	10	0.016	32	10	1	0.5
5/2	Tunnel	3.238	2,443	416	0.947	1,116	273	2.75	1
	MRS	64.0	*	*	*	*	*	1	0
	MMRS	1.062	1,241	287	0.997	1,195	287	3	1
6/3	Tunnel	12.915	7,325	1,328	4.435	3,823	848	3.25	1
	MRS	3.516	2,861	784	3.485	2,861	784	3	1
	MMRS	4.024	3,443	726	3.231	2,647	726	3	1
7/4	Tunnel	20.450	4,881	1,371	1.91	1,126	376	4.25	1
	MRS	3.391	*	*	*	*	*	6	0
	MMRS	4.335	3,076	457	2.289	1,369	457	2	1
8/5	Tunnel	11.885	7,540	1,122	9.32	6,471	881	2	1
	MRS	8.107	*	*	*	*	*	1	0
	MMRS	11.925	9,458	171	2.112	1,506	171	1	0
9/8	Tunnel	45.474	19,366	2,370	35.644	16,138	2,011	2.5	1
	MRS	38.091	17,229	2,143	38.091	17,229	2,143	2	1
	MMRS	45.535	22,193	1,771	30.757	14,126	1,771	1	0
10/10	Tunnel	68.22	23,982	3,272	67.283	22,191	3,135	2.5	1
	MRS	192.0	*	*	*	*	*	1	0
	MMRS	68.26	25,966	2,913	62.47	23,093	2,913	1	0
11/2	Tunnel	4.364	2,613	322	0.762	736	158	2.5	0.5
	MRS	6.308	6,851	876	6.308	6,851	876	5	0
	MMRS	1.792	1,867	250	1.406	1,441	250	4	1
12/3	Tunnel	12.378	6,955	754	3.927	3,142	479	4.25	0.75
	MRS	13.291	10,566	1,652	13.227	10,566	1,652	9	0
	MMRS	2.975	2,316	359	2.139	2,316	359	3	1
13/4	Tunnel	8.35	3,861	588	3.076	1,863	390	3.75	0.75
	MRS	9.851	6,659	740	9.821	6,659	740	2	0
	MMRS	8.376	6,234	273	2.294	1,419	273	2	0
14/5	Tunnel	28.33	10,715	1,507	7.249	3,565	797	5.5	0.75
	MRS	51.707	28,347	4,002	51.712	28,347	4,002	7	0
	MMRS	28.362	17,339	1,098	11.150	5,746	1,098	3	0
15/6	Tunnel	33.173	12,786	1,777	17.282	7,839	1,329	3.5	1
	MRS	41.065	19,301	2,784	41.028	19,301	2,784	2	0
	MMRS	33.231	18,985	132	1.263	479	132	2	0
16/7	Tunnel	71.981	16,063	2,792	15.350	6,142	1,013	7.5	0.75
	MRS	92.615	38,483	5,411	92.546	38,483	5,411	8	0
	MMRS	72.027	36,195	435	5.977	2,132	435	2	0

* Failure in convergence.

TABLE 2
Results for the tunneling algorithm coupled with alternate local minimization procedures.

Ex/Dim.	Minimization phase	t_{av}	t_{min}	Total no. of minimizations	Globals found
1/2	Ord. grad.	87.045	2.113	17	17
	Conj. grad.	105.474	4.283	19.75	17.25
	Newton	128.656	3.210	18	17
2/2	Ord. grad.	8.478	1.054	3	1
	Conj. grad.	4.656	0.064	6.75	1
	Newton	8.173	2.157	7	1
3/2	Ord. grad.	5.984	1.409	3	1
	Conj. grad.	5.944	0.446	4.75	1
	Newton	10.466	1.699	7	1
4/2	Ord. grad.	1.984	0.033	2	2
	Conj. grad.	0.586	0.070	2	2
	Newton	0.43	0.061	3.5	2
5/2	Ord. grad.	3.283	0.947	2.75	1
	Conj. grad.	1.052	0.091	3.75	1
	Newton	0.327	0.110	2.125	1
6/3	Ord. grad.	12.915	4.435	3.25	1
	Conj. grad.	7.218	0.187	3	1
	Newton	5.863	0.377	2.75	1
7/4	Ord. grad.	20.45	1.910	4.25	1
	Conj. grad.	11.782	0.323	3.125	1
	Newton	5.29	3.480	2	1
8/5	Ord. grad.	11.885	9.320	2	1
	Conj. grad.	15.850	1.593	5.25	1
	Newton	46.195	6.829	11.25	1
9/8	Ord. grad.	45.474	35.644	2.5	1
	Conj. grad.	24.773	6.001	4.75	1
	Newton	157.830	78.941	15	1
10/10	Ord. grad.	68.220	67.283	2.5	1
	Conj. grad.	118.737	7.890	11.25	1
	Newton	230.381	208.570	6.75	1

this means that the ordinary gradient method was not easily trapped by local minimizers as it probably occurred to the alternate procedures. (Obviously, this depends on the one-dimensional search being used.) However, this behavior has not yet been satisfactorily explained.

6. Final comments. Based on the material previously presented we conclude that the Tunneling Algorithm has the following characteristics that make it very attractive for finding global minima.

a) It is superior to several other methods previously reported since generally, it locates the global minimizer(s) faster than do the other methods. It should be pointed out that even in the case when the tunneling algorithm is not highly efficient (for several examples) it still locates them.

b) It was experimentally observed that the efficiency ∩ the tunneling algorithm is not affected by the density of relative (nonglobal) minimizers, while the other methods tested are strongly affected, in a negative direction, by this property of the objective function.

c) When comparing different local minimization methods coupled to the tunneling algorithm, no one of these showed to be more efficient than the others. However, it should be pointed out that when the tunneling algorithm was coupled to the ordinary gradient method, this version required the least amount of minimization phases.

Appendix I.

Determination of the tunneling function. In this appendix we present the necessary steps to determine the actual parameters involved in the definition of $T(x)$, namely

$$[l, (x_i^*, i=1, 2, \cdots, l), (\eta_i, i=1, 2, \cdots, l), x_m, \lambda_0, f^*]$$

where $T(x)$ was defined as

$$T(x)=\frac{f(x)-f^*}{\{\prod_{i=1}^{l}[(x-x_i^*)'(x-x_i^*)]^{\eta_i}\}[(x-x_m)'(x-x_m)]^{\lambda_0}}. \tag{A.0}$$

Step 1. *Determination of* f^*. Once any local minimum has been found at x^*, we want the tunneling algorithm to ignore all the local minima whose function value are higher than $f(x^*)$, since they are irrelevant for approaching the global minimizer. This objective can be achieved by letting f^* be the lowest function value obtained so far. With this selection of f^* we accomplish our objective of tunneling below irrelevant local minima even if we do not know how many they are nor their location.

Step 2. *Determination of* η_i *and* x_i^*, $i=1, 2, \cdots, l$. To clearly illustrate this step, let us assume that $l=1$ that is the algorithm has found only one local minimum at x_1^*, with a function value $f^*=f(x_1^*)$. We employ Equation (A.0) with $\lambda_0=0$, thus the tunneling function simplifies to (λ_1 replacing η_1)

$$T(x)=\frac{f(x)-f^*}{[(x-x_1^*)'(x-x_1^*)]^{\lambda_1}} \tag{A.1}$$

where $x=x_1^*+\varepsilon$. The correct value of η_1 is found iteratively starting from $\lambda_1=1$, until the following descent property holds.

$$T_x'(x)\Delta x<0 \tag{A.2}$$

provided

$$\varepsilon'\Delta x>0 \tag{A.2a}$$

where ε is a random vector with $\|\varepsilon\| \ll 1$, Δx is the displacement produced by the tunneling algorithm using the trial value of λ_1 in (A.2). It can easily be proved that (A.2) and (A.2a) are simultaneously satisfied provided $T(x)$ has a pole at $x=x_1^*$.

If for a given λ_1, inequalities (A.2) are not satisfied, increase λ_1 by $\Delta\lambda_1$ until the above descent property is satisfied. For subsequent steps the actual value of η_1 is determined according to

$$\eta_1=\begin{cases}0 & \text{if } \gamma \geqq 1+\varepsilon_2,\\ \lambda_1 & \text{if } \gamma \leqq 1-\varepsilon_2,\\ (\lambda_1/2)[1+(1-\gamma)/\varepsilon_2] & \text{if } 1-\varepsilon_2 \leqq \gamma \leqq 1+\varepsilon_2\end{cases} \tag{A.3}$$

where $\gamma=\|x-x^*\|$ and ε_2 is a small prescribed number. Note that Equation (A.3) is

a ramp function which continuously switches the denominator in Equation (A.1), when crossing the unit circle centered at x_1^*. This switching procedure is necessary because for some functions, as the tunneling algorithm iterates, the denominator in Equation (A.1) might become very large when $\|x-x_1^*\|>1$, forcing the tunneling function to become very flat and close to zero, thus slowing down the convergence of the tunneling algorithm.

Let us now consider the case of $l>1$, that is, when the function has multiple local minima at the level f^*, and l successive applications of the minimization phase with $(l-1)$ tunneling phases have identified the points x_i^*, $i=1,2,\cdots,l-1$ as local minimizers of $f(x)$ at the level f^*. As each x_i^*, $i=1,2,\cdots,l-1$ is found and having computed η_i, $i=1,2,\cdots,l-1$, η_l is computed employing the above iterative procedure using the function

$$T(x)=\frac{f(x)-f^*}{\prod_{i=1}^{l-1}[(x-x_i^*)'((x-x_i^*)]^{\eta_i}[(x-x_l^*)'(x-x_l^*)]^{\lambda_l}}. \tag{A.4}$$

The actual value of η_l is determined as in (A.3), that is

$$\eta_l=\begin{cases}0 & \text{if } \gamma_l\geqq 1+\varepsilon_2,\\ \lambda_l & \text{if } \gamma_l\leqq 1-\varepsilon_2,\\ (\lambda_l/2)[1+(1-\gamma_l)/\varepsilon_2] & \text{if } 1-\varepsilon_2\leqq\gamma_l\leqq 1+\varepsilon_2\end{cases} \tag{A.5}$$

where $\gamma_l=\|x-x^*\|$.

Remark. The value of l is reset to $l=1$ whenever the new local minimum found has a lower value than the previous one.

Step 3. *Determination of* x_m *and* λ_0. For the first iteration of each tunneling phase we set

$$x_m=x_l^* \quad \text{and} \quad \lambda_0=0. \tag{A.6}$$

For subsequent iterations within the tunneling phase the position x_m and pole strength λ_0 of the movable pole are automatically computed in order to cancel out any undesirable relative minimum that the function $T(x)$ might have. Once any of these minima is detected we proceed as follows.

Let x be the present point and $\hat{x}$ the previous point generated by the tunneling algorithm. Then the position of x_m is determined as follows

$$x_m=\begin{cases}\hat{x} & \text{if } \|\hat{x}-x\|<1,\\ \xi\hat{x}+(1-\xi)x & \text{if } \|\hat{x}-x\|\geqq 1\end{cases} \tag{A.7}$$

where $0<\xi\leqq 1$ is chosen such that $\|x-x_m\|<1$. We note that the movable pole thus located will always be within the unit circle centered at the current position x of the zero finding algorithm, so we are free to choose the pole strength as large as it is required to cancel out any undesirable minimum that $T(x)$ might have in the neighborhood of x. To achieve this, let λ_0 be the pole strength value employed when moving from the previous point $\hat{x}$ to x: let Δx be the displacement produced by the tunneling algorithm leading from x to the new point $\tilde{x}$ and consider the inner product

$$u=(x-\hat{x})'\Delta x. \tag{A.8}$$

If $u>0$, we can retain the pole strength while moving from x to $\tilde{x}$. If $u<0$, the value of λ_0 is no longer appropriate, and a new λ_0 is found iteratively starting from $\lambda_0=\hat{\lambda}_0$ and increasing it by $\Delta\lambda_0$ until $u>0$.

Once the tunneling algorithm has moved away from the local minimum of $T(x)$

which motivated the increase of the pole strength, one could return to a simpler tunneling function with a simpler geometry, by resetting the movable pole strength λ_0 to zero.

A heuristic rule to detect when the tunneling algorithm has left the undesirable neighborhood of the local minimum of $T(x)$ is as follows. Compute the displacement that would be produced by the tunneling algorithm, for two values of λ_0, namely, for the current value of λ_0 we compute $\Delta x(\lambda_0)$; for $\lambda_0 = 0$ we compute $\Delta x(0)$. Next, we measure the angle between these two displacements and update the value of the pole strength λ_0 according to the following rule:

$$\lambda_0 = \begin{cases} 0 & \text{if } \Delta x'(0)\Delta x(\lambda_0) > 0, \\ \lambda_0 & \text{if } \Delta x'(0)\Delta x(\lambda_0) \leqq 0. \end{cases} \tag{A.9}$$

Step 4. *Stopping condition for the tunneling phase.*

Case 1. Recall that the tunneling algorithm should be stopped whenever $T(x) \leqq 0$. In the actual implementation of this phase this condition has been replaced by

$$T(x) \leqq \varepsilon_3. \tag{A.10}$$

We note that any x satisfying (A.10) is an acceptable starting point for the next minimization phase.

Case 2. This case occurs when for a given ε-vector, in a given number of iterations, N_s, the zero finding algorithm fails to locate $x^0 \in \Omega$ such that $T(x^0) \leqq \varepsilon_3$. If this is the case we are in the position of selecting another ε-vector, different from the previous one, and restart the tunneling phase for this new ε-vector, until satisfaction of inequality (A.10) is achieved; this procedure is repeated N_ε times.

Finally, without changing any of the parameters of the tunneling function $T(x)$, we initiate the tunneling phase from any starting point selected at random $x \in \Omega$, that is, no longer within an ε-neighborhood of the last minimum found; this procedure is repeated N_R times.

The search for any point x^0 such that $T(x^0) \leqq 0$ was implemented using the so-called Restoration Algorithm (Miele [8]), as described in § 2.3.4, to produce the displacements Δx. The values assigned to the parameters involved in this phase were the following:

The convergence criteria were chosen as

$$\varepsilon_3 = 10^{-3};$$

the η-switching parameter was fixed at

$$\varepsilon_2 = 10^{-5};$$

and the nonconvergence criteria were the following:

$$: \lambda_{\max} = 5$$

$$: \lambda_0^* = 1$$

$$: \beta_R = 2^{-20}$$

Number of bisections: $N_b = 20$

Number of search steps: $N_s = 100$

Number of ε-vectors: $N_\varepsilon = 2n$

Number of final random vectors: $N_R = 2n$

where n denotes the dimensionality of the problem.

REFERENCES

[1] R. S. ANDERSSEN AND P. BLOOMFIELD, *Properties of the random search in global optimization*, J. Optim. Theory Appl., 16 (1975), pp. 383–398.

[2] M. AVRIEL, *Nonlinear Programming Analysis and Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[3] F. H. BRANNIN, JR., *Widely convergent method for finding multiple solutions of simultaneous nonlinear equations*, IBM J. Res. Dev. (September 1972), pp. 504–522.

[4] L. DIXON AND G. P. SZEGO, eds., *Towards Global Optimization*, North-Holland, Amsterdam, 1975.

[5] A. A. GOLDSTEIN AND J. F. PRICE, *On descent from local minima*, Math. Comp., 25 (1971), pp. 569–574.

[6] J. HARDY, *An implemented extension of Brannin's method*, in Towards Global Optimization, North-Holland, Amsterdam, 1975.

[7] D. M. HIMMELBLAU, *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.

[8] A. MIELE, H. Y. HUANG AND J. C. HEIDEMAN, *Sequential gradient-restoration algorithm for the minimization of constrained functions—ordinary and conjugate gradient versions*, J. Optim. Theory Appl., 4 (1969), pp. 213–243.

[9] A. MONTALVO, *Desarrollo de un nuevo algoritmo para la minimización global de funciones*, Ph.D. thesis, School of Engineering, National Autonomous University of Mexico, 1979.

[10] B. O. SHUBERT, *A sequential method seeking the global maximum of a function*, SIAM J. Numer. Anal., 9 (1972), pp. 379–388.