MATEMATIKA, 2013, Volume 29, Number 1c, 221-231
© Department of Mathematical Sciences, UTM.

Classical Tunneling Function versus Exponential Tunneling Function in Tunneling Algorithm

[1]Zahra Roshan Zamir and [2]Rohanin Ahmad
[1,2]Department of Mathematics, Faculty of Science, University Teknologi Malaysia
81310 UTM Johor Bahru, Malaysia
e-mail: [1]z.roshanzamir@gmail.com, [2]rohanin@utm.my

Abstract Global optimization is an active research area due to its ability to provide the best set of parameters to optimize an objective function in the area of interest. Most of real life problems are large-scale and nonlinear therefore it is felt necessary to find the global solution for large-scale and nonlinear optimization problems. Finding global minimizer of a function defined in an *n*-dimensional linear space is one of the most interesting areas in nonlinear programming. This paper considers the performance of the classical tunneling function in comparison with the exponential tunneling function in the tunneling algorithm. The proposed algorithm is applied to multivariable and nonlinear functions. For both tunneling functions, the tunneling algorithm was able to find the global minimizer.

Keywords Global Optimization; Classical Tunneling Function; Exponential Tunneling Function; Tunneling Algorithm; Unconstrained Minimizations.

2010 Mathematics Subject Classification 90C26, 90C30

1 Introduction

Many problems in engineering, physics, economics, and other fields are reduced to global minimization with many local minimizers. Mathematically the problem is formulated as:

$$f^* = \min_{x\in D} f(x),\ f : R^N \to R, D \subseteq R^N,$$

where f is a nonlinear function of continuous variables and D is a feasible region and N is the number of variables. Besides the global minimum f^*, one or all global minimizers x^* such that $x^* : f(x^*) = f^*$ should be found. No assumptions on unimodality are included into formulation of the problem. Global optimization problems are classified difficult in the sense of the algorithmic complexity theory. Therefore, global optimization algorithms are computationally intensive, and solution time crucially depends on the dimensionality of a problem.

A point x^* is a local minimum point of the function f if

$$f(x^*) \le f(x) \text{ for } x \in N,$$

where N is a neighborhood of x^*. A local minimum point can be found by using local optimization methods such as steepest descent method, conjugate gradient method and Newton's method. Without additional information, one cannot say if the local minimum is global. Global optimization methods not only try to find a good function value fast, but also try to explore the whole feasible region by evaluating function values at sampling points or investigating sub-regions of the feasible region. A major drawback of any local optimization algorithm is that, it does not search the whole domain, which is evidently necessary in global optimization. Global optimization methods are important in many applications because even better solution can translate into large saving in time, money and resources. Most of real life problems are large-scale and nonlinear therefore it is felt necessary to find the global solution for these problems. Finding global minimizer of a function is one of the most interesting areas in nonlinear problems [1]. The classical global optimization techniques for example, “the hill climbing method”, “multiple random start method” and “the function modification method” are applicable only for low dimensionality problems [1]. Finding the global minimizer of a function with higher dimensionality is a lot more challenging and time consuming. The main drawback of the classical global minimization methods is premature convergence because they get stuck in the local minimum. These methods always get stuck in the local minimum since their searching efficiency is affected by the initial points and the topology of the surface associated with the objective function [2].

In order to improve the efficiency of the classical techniques Levy and Montalvo (1977) presented the tunneling algorithm (TA) for unconstrained minimization problems in Dundee conference then the unconstrained minimization problem expanded to constrained minimization problems by Levy and Gomez (1985) [3]. The name of the algorithm describes its goal to tunnel from one valley of the objective function to another to find a sequence of local minima with decreasing values.

Tunneling algorithm is composed of a sequence of cycles, where each cycle has two phases: a minimization phase and a tunneling phase. In the first phase, we start from a given initial point and use any local minimization method to find a local minimizer x^* of $f(x)$. In the second phase, a classical tunneling function or exponential tunneling function is used as a transformation function at x^* and minimize the classical or exponential tunneling function in order to identify a point x^0 with $f(x^0) < f(x^*)$. We can then use x^0 as the initial point for the next minimization phase, hence we can find a better local minimizer x^{**} of $f(x)$ with $f(x^{**}) < f(x^*)$. This process repeats until the local minimization of the classical or exponential tunneling function does not yield a better point. The current local minimum will then be taken as a global minimizer of $f(x)$. Existing global optimization algorithms can be divided into deterministic and stochastic ones. In either case, only approximate solutions are obtained. This paper considers the tunneling algorithm as a deterministic method of global optimization. In this paper both tunneling functions namely classical tunneling function and exponential tunneling function will be employed to compare the performance of them in the tunneling algorithm.

2 Fundamental of Tunneling Algorithm

The tunneling algorithm consists of two phases, a minimization phase and a tunneling phase. At the minimization phase for a given starting point x^{00} the local minimum x^* is found by using any minimization algorithm with a descent property on $f(x)$. The tunneling phase starts from x^* that was obtained in the minimization phase and then proceeds to find a point $x^0 \in \Omega$ such that

$$f(x^0) \le f(x^*), x^0 \ne x^*, \Omega = \{x \in R^n : a \le x \le b\}.$$

In order to move away from x^*, auxiliary functions $T(x)$ (classical tunneling function or exponential tunneling function) are defined as follows:

$$T_c(x) = \frac{f(x) - f^*}{\| x - x^* \|^{2\lambda^*}}, \tag{1}$$

or

$$T_e(x) = (f(x) - f^*)\exp(\frac{\lambda^*}{\| x - x^* \|}). \tag{2}$$

The functions T_c and T_e are called the classical tunneling function (CTF) and the exponential tunneling function (ETF) respectively. Both of them create a pole placed at x^* with pole strength λ^*. This pole is needed to move away from the current local minimizer. When the x^0 in the tunneling phase is found such that:

$$f(x^0) \le f(x^*), x^0 \ne x^*,$$

then x^0 will serve as the initial point for the new minimization phase that will locate a new minimizer for the next iteration. On the other hand, as we are trying to find a point x^0 in another valley with less or equal value than f^* we need to solve the following inequality

$$T_{c,e}(x) = f(x) - f(x^*) \le 0, \tag{3}$$

solving problem (3) consists in finding x^0 such that

$$T_c(x^0) \le 0 \quad \text{or} \quad T_e(x^0) \le 0. \tag{4}$$

The cycle of minimization phase and tunneling phase is repeated until it finds the global minimizer [4]. If a large value of λ^* is taken, both tunneling functions will be smoother and the danger of encountering critical points during the research will be reduces therefore the search for the point x^0 in the inequality (4) will be more expensive for large value of λ^*. Thus it is better to choose pole strength λ^* with small value [5].

Assume that there is a continuous function $f(x)$ with the $x - f(x)$ relation plotted in Figure1, the tunneling method starts with an initial point $x_1$and uses a local minimization technique such as gradient descent or Newton's method to find the nearest local minimum 'a' in the minimization phase. The tunneling phase starts from point 'a' and finds the tunneling point x_2. The tunneling point x_2 is used as the starting point for the next minimization phase in the second cycle and in this phase the second lowest minimum point 'c' will be located. Here tunneling phase starts from minimum point 'c' which was obtained in the previous minimization phase and finds the tunneling point x_3. The tunneling point x_3 will be an initial point for the next minimization phase. The minimization phase starts from x_3 and finds the global minimum at 'e'. Generally, the tunneling algorithm will terminate after a specified maximum number of iterations. All of the above procedures are shown in Figure 1.

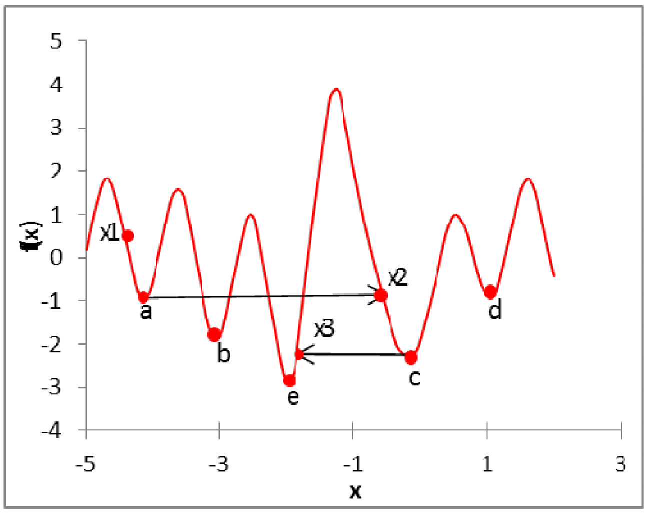


Figure 1 Finding the global minimum of the non-linear function by using tunneling algorithm

3 Problem Formulation

The optimization problem can be stated as $f(x): D \rightarrow R$, where x is a state vector having a finite dimension in $D \subseteq R^n$, D is called the domain or the search space, R represents the range and $f(x)$ will denote the objective function. In this work one of the objectives is to find the global minimizer x_G with its minimum value $f(x_G)$:

$$\min_{x \in B} f(x),\ B = \{x \mid x_{\min} \le x \le x_{\max}\}, f: R^n \rightarrow R, x \in R^n, f \in C^2,$$

n denotes the dimensionality of the problem. In this work $f(x)$ is a twice continuously differentiable function on the set B. It could have many local and global minima. The tunneling algorithm can be used for unconstrained and constrained problem but this paper confined to the unconstrained problem since it is appeared that the local minimum problem for the unconstrained problem is already difficult enough. Therefore in this paper tunneling method applies for solving nonlinear unconstrained minimization problem on $f(x)$ with continuous first and second derivatives and x is an n-vector.

4 The Global Optimization Method

The basic idea of tunneling is to tunnel from one valley of the objective function to another to find a sequence of local minima with decreasing function values:

$$\{f(x_1) \ge f(x_2) \ge \dots \ge f(x_G)\} \tag{5}$$

where x_G is the global minimum of $f(x)$.

4.1 Minimization Phase

The problem here is to find a local bounded minimum:

$$\text{Find} \quad x_m = \arg(\min_{x \in B} f(x)) \tag{6}$$

To perform minimization phase any minimization algorithm with bounds on the variables and descent property on $f(x)$ can be used. We used Quasi Newton method with BFGS hessian update to find the local solution.

4.2 Tunneling Phase

Once a local minimum has been found, we want to find a point in another valley therefore we need to solve the following inequality problem:

$$\text{Find} \quad x_m^* \in B \text{ such that } f(x_m^*) \le f(x_m) \tag{7}$$

To be able to find a point x_m^* in another valley with less or equal value than $f(x_m)$ we first create classical tunneling function $T_c(x)$ or exponential tunneling function $T_e(x)$ such that, if $f(x) \le f(x_m)$ then $T_c(x) \le 0$ or $T_e(x) \le 0$. Whenever tunneling phase starts to tunnel from one valley to another using gradient-type method, it is necessary to destroy the minimum, placing a pole at the minimum point x_m and then to start tunneling from an initial point nears the minimum. In this paper both tunneling functions namely classical tunneling function (CTF) and exponential tunneling function (ETF) will be employed to compare the performance of them in the tunneling algorithm. We need to take descent directions to solve the inequality $T_c(x_m^*) \le 0$ or $T_e(x_m^*) \le 0$. Thus we use the same algorithm used to solve equation (6).

4.3 Stopping Conditions

The algorithm stops when any of the following criteria is satisfied:

- In the tunneling phase when $T_{c,e}(x) \le 0$ or a maximum number of iterations is reached and the last minimum found is the putative global minimum.
- The stopping criterion is considered as $\| g(x_m) \| \le \varepsilon$ for local minimization techniques in the minimization phase. Here g is the gradient of the objective function. The value $\varepsilon = 10^{-4}$ was chosen in the minimization phase.

5 Numerical Experiment

The main objective of this work is to compare the performance of the exponential tunneling function with the classical tunneling function in the tunneling algorithm when tunneling algorithm locates the global minimizer x_G. Making the sequence of decreasing function values is another objective of using tunneling algorithm in this paper. In order to facilitate a comparison of the performance of the tunneling algorithm when using both tunneling functions (classical and exponential tunneling functions) in the tunneling phase, three test functions are taken as objective functions. These are then compared with the results available in literatures [6,7] in terms of the obtained global minimizer and the computed value of the objective function at that point. The results are presented below in Table 1. Whenever the minimum point in the minimization phase is found then tunneling phase starts from an initial point near the minimum point to tunnel from one valley to another. That is why we need a small perturbation $\in$ such that $x_m^0 = x_m + \in$, where x_m is founded at the minimization phase. Therefore tunneling phase starts from x_m^0. Small perturbations $\in = 0.01$ and $\delta = 0.1$ are selected for classical and exponential tunneling function respectively for all test functions. Also, the pole strength λ^* is selected to be 1. Table 1 gives the local solutions obtained in the minimization phase in every cycle, the new initial points obtained in the tunneling phase, function values for every local solution and the number of iterations for each phase in every cycle.

5.1 Test Function 1

It is an *n*-dimensional function. In this test function $n = 2$ and initial solution $x^0 = [4.0, 6.4]$ were selected.

$$f(x) = 1/2\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i) \tag{8}$$

The tunneling algorithm started from initial solution $x^0 = [4.0, 6.4]$ with the function value f=537.181 when we used CTF in the tunneling phase. It is clear from Table 1 that in the minimization phase the local solution $x^{*1} = [2.747, -2.904]$ was found after seven iterations in the first cycle which is an initial point for the tunneling phase to tunnel from x^{*1} to another valley by using CTF. After eight iterations the tunneling phase offered the new tunneling point $x^{01} = [-2.747, -2.904]$. In the second cycle the algorithm started from the new initial point $x^{01} = [-2.747, -2.904]$ and the local solution $x^{*2} = [-2.904, -2.904]$ was found by the minimization phase after two iterations. After that the tunneling phase did not find any new starting point, hence the last local solution was accepted as the global optimum and the algorithm terminated. The sequence of decreasing function values for this test function when we use CTF in the tunneling phase is as follow:

$$\{537.181, -64.1956, -77.9306, -78.3323\}.$$

The last value in this sequence considers as the value of objective function for the global minimizer. Therefore the point $x^{*2} = [-2.904, -2.904]$ is a global minimizer with the corresponding minimum value $f = -78.3323$ which is found after two cycles.

When we use ETF in the tunneling phase, the algorithm started from the initial point $x^0 = [4.0, 6.4]$ with the function value $f = 537.181$. The local solution $x^{*1} = [2.747, -2.904]$ was found by the minimization phase in the cycle 1 after seven iterations, after which the tunneling phase offered the new initial point $x^{01} = [-2.891, -2.904]$ after five iterations.

In the second cycle the algorithm started from the new initial point $x^{01}=[-2.891,-2.904]$ and the local solution $x^{*2}=[-2.904,-2.904]$ was found by the minimization phase after two iterations. After that the tunneling phase did not find any new initial point, and then the last local solution was considered as the global minimizer. The sequence of decreasing function values for this test function when we use ETF in the tunneling phase is as follow:

$$\{537.181,-64.1956,-78.3296,-78.3323\}$$

It can be seen from Table 1 that when the ETF was used in the tunneling phase, the tunneling algorithm can find the global minimizer $x^{*2}=[-2.904,-2.904]$ which is the same as the global minimizer that was found by using CTF in the tunneling phase. When the ETF is used in the tunneling phase, the function value $f(x^{01})=-78.3296$ is obtained after five iterations. This function value is less than the function value $f(x^{01})=-77.9306$ which was found after eight iterations in the tunneling phase when the CTF was used, that is why the ETF is more efficient than CTF. It is obvious that, for finding the tunneling point x^{01}, the ETF needs less iterations than CTF.

5.2 Test Function 2: Rosenbrock Function

The Rosenbrock function is selected for the second test function. This function also known as Banana function.

$$f(x)=\sum_{i=1}^{n-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2] \tag{9}$$

Search domain for this test problem is usually $-5\le x_i\le 10, i=1,2,...,n$ and *n* denotes the dimensionality of the problem. For this problem *n*=4 and the initial point $x^0=[-4,-4,0,2]$ were selected.

The tunneling algorithm started from $x^0=[-4,-4,0,2]$ with the function value $f=66051$ when we used CTF in the tunneling phase. It is clear from Table 1 that after fifty-three iterations, minimization phase can find the minimum point $x^{*1}=[0.999,0.999,0.997,0.994]$ with the function value $f=2.1101e-4$. This point is an initial point for the tunneling phase. After that the tunneling phase, started from local solution $x^{*1}=[0.999,0.999,0.997,0.994]$ and found the tunneling point $x^{01}=[0.999,0.999,0.997,0.994]$ after thirty-six iterations in the first cycle. It is obvious that the tunneling phase did not change the minimum point x^{*1}. In the second cycle, the algorithm started from the new initial point (tunneling point) $x^{01}=[0.999,0.999,0.997,0.994]$, and the local solution $x^{*2}=[1,1,1,1]$ was found by the minimization phase after nineteen iterations. The tunneling phase could not offer any tunneling point as new initial point for the next phase then the last local solution $x^{*2}=[1,1,1,1]$ was accepted as the a global minimizer and the algorithm terminated.

When we used ETF in the tunneling phase, the tunneling algorithm started from initial solution $x^0=[-4,-4,0,2]$ with the function value $f=66051$, and the local solution $x^{*1}=[0.999,0.999,0.997,0.994]$ was found after fifty-three iterations by the minimization phase in the first cycle, after that the tunneling phase found the new initial point $x^{01}=[1,1,1,1]$ after eighteen iterations.

In the second cycle, the algorithm started from the new initial point $x^{01} = [1,1,1,1]$, and then the local solution $x^{*2} = [1,1,1,1]$ was found by the minimization phase after one iteration. After that the tunneling phase did not offer new initial point and the point x^{*2} is the same as the reported global minimizer with the function value f=0, thus the algorithm terminated.

The tunneling point $x^{01} = [1,1,1,1]$ was found after eighteen iterations when the ETF was used in the tunneling phase. This point is the same as the reported global minimizer. But the tunneling point $x^{01} = [0.999, 0.999, 0.997, 0.994]$ was found after thirty-three iterations when the CTF was used in the tunneling phase. This point is so close to the reported global minimizer. Therefore the ETF is more efficient than CTF in the tunneling phase. It can be seen that for finding the tunneling point x^{01}, the ETF needs less iterations than CTF to locate the tunneling point x^{01}. The tunneling algorithm can locate the global minimizer $x^{*2} = [1,1,1,1]$ with the function value $f = 0$ after two cycles for both tunneling functions CTF and ETF. It is obvious that the exponential tunneling function can create a pole at a local minimizer independent of the precision of the local minimum found.

The sequences of decreasing function values for this test function when we use CTF and ETF are $\{66051, 2.1101e-4, 2.1101e-4, 0\}$ and $\{66051, 2.1101e-4, 0, 0\}$ respectively. The last values of both sequences are considered as the function value for the global minimizer $x^{*2} = [1,1,1,1]$.

5.3 Test Function 3

The third test function is stated as below:

$$f(x) = \frac{\pi}{n}\{k\sin^2(\pi x_1) + \sum_{i=1}^{n-1}[(x_i - A)^2$$
$$(1 + k\sin^2(\pi x_{i+1}))] + (x_n - A)^2\}, \tag{10}$$
$$-10 \le x_i \le 10, i = 1,2,...,n,$$

where the constants k and A were fixed at 10 and 1 respectively and n denotes the dimensionality of the problem. For this test problem n=8 and the initial solution $x^0 = [8,\ 8,\ 8,\ 8,\ 8,\ 8,\ 8,\ 8]$ were selected.

When we used CTF in the tunneling phase the algorithm started from the initial point $x^0 = [8,\ 8,\ 8,\ 8,\ 8,\ 8,\ 8,\ 8]$ with the function value $f = 153.9380$ and the local solution $x^{*1} = [0.972,1,1,1,1,1,1,1]$ was found by the minimization phase in the initial cycle after two iterations, after which the tunneling phase offered the initial point $x^{01} = [0.972,1,1,1,1,1,1,1]$ after twelve iterations which is the same as x^{*1}.

In the second cycle the algorithm started from the new initial point $x^{01} = [0.972,1,1,1,1,1,1,1]$, and the local solution $x^{*2} = [1,1,...,1]$ was found by the minimization phase after three iterations. After that, the tunneling phase did not find any new starting point, hence the last local solution was accepted as the global optimal solution and the algorithm terminated. When we use ETF in the tunneling phase, the algorithm started from the initial point $x^0 = [8,\ 8,\ 8,\ 8,\ 8,\ 8,\ 8,\ 8]$ with the function value $f = 153.9380$ and the local solution $x^{*1} = [0.972,1,1,1,1,1,1,1]$ was found by the minimization phase in the first cycle after two iterations. Then the tunneling phase found the new initial point x^{01} after twenty–

two iterations in the first cycle with the function value $f = 0.0210$ such that:

$$x^{01} = [0.978, 1.016, 0.994, 1.006, 0.934, 1.017, 1.01, 0.997].$$

Although tunneling phase by using ETF found the point x^{01} after twenty-two iterations which is more than the number of iterations for finding x^{01} by using CTF but it found the new initial point that is different from local solution x^{*1}. It can be seen that when CTF used in the tunneling phase, the tunneling phase did not offer the new initial point for the next minimization phase. It found $x^{01} = [0.972, 1, 1, 1, 1, 1, 1, 1]$ that is the same as the local solution x^{*1}. Therefore ETF is more efficient than CTF because it created a point that is different from local solution x^{*1}. The sequences of decreasing function values for this test function when we use CTF and ETF are as follows:

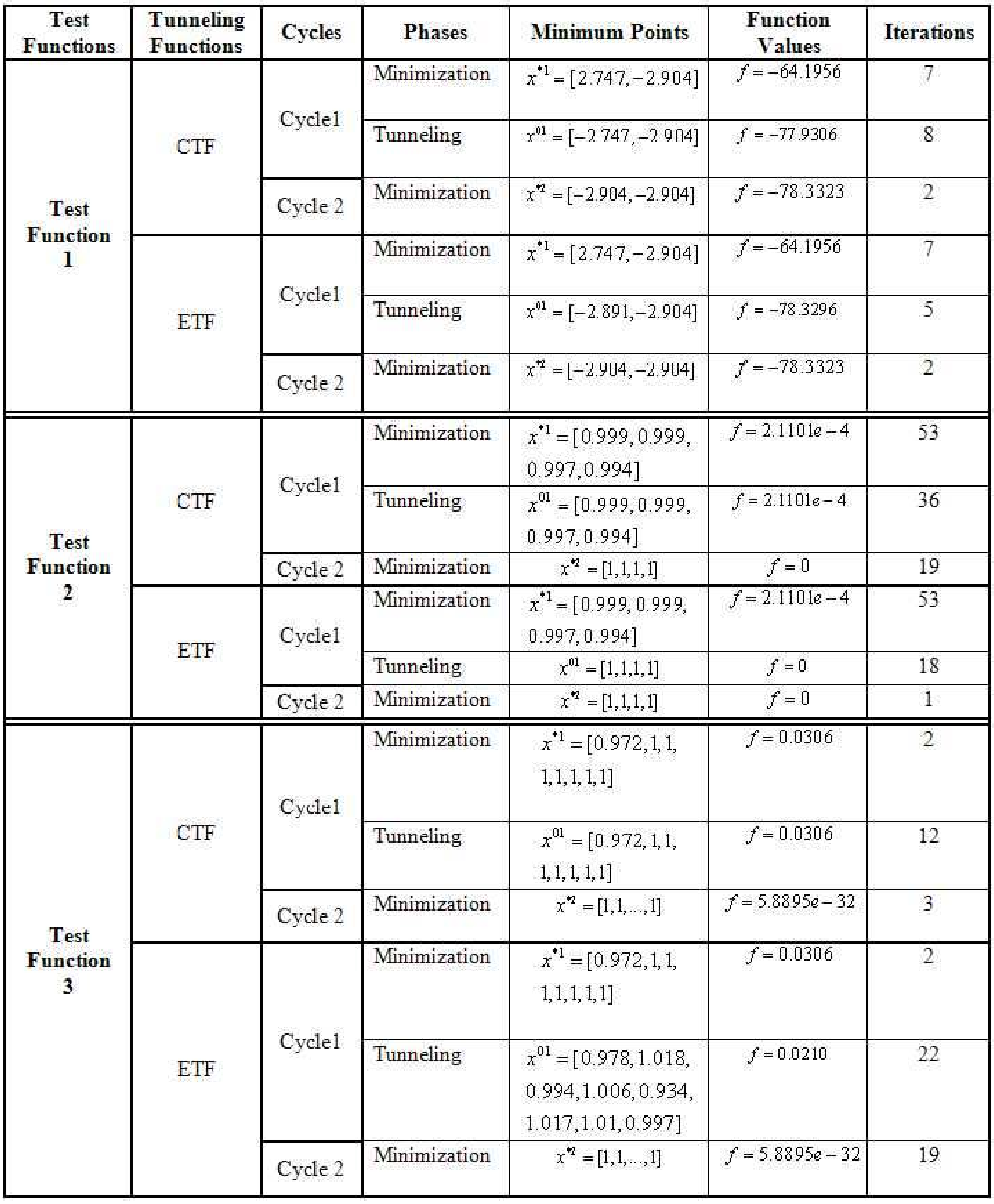
$$\{153.9380, 0.0306, 0.0306, 5.8895e-32\} \text{ for CTF}$$

and

$$\{153.9380, 0.0306, 0.0210, 5.8895e-32\} \text{ for ETF.}$$

The last values for both sequences are considered as the global minimum. At the end the tunneling algorithm can locate the global minimizer $x^{*2} = [1, 1, ..., 1]$ with the corresponding minimum value $f(x^{*2}) = 5.8895e-32$ when both CTF and ETF were used in the tunneling phase.

Table 1 Simulation Results for three Test Functions when both tunneling functions namely Classical Tunneling function (CTF) and Exponential Tunneling Function (ETF) are employed in the Tunneling Phase



Test Functions	Tunneling Functions	Cycles	Phases	Minimum Points	Function Values	Iterations
Test Function 1	CTF	Cycle1	Minimization	$x^{*1} = [2.747, -2.904]$	$f = -64.1956$	7
			Tunneling	$x^{01} = [-2.747, -2.904]$	$f = -77.9306$	8
		Cycle 2	Minimization	$x^{*2} = [-2.904, -2.904]$	$f = -78.3323$	2
	ETF	Cycle1	Minimization	$x^{*1} = [2.747, -2.904]$	$f = -64.1956$	7
			Tunneling	$x^{01} = [-2.891, -2.904]$	$f = -78.3296$	5
		Cycle 2	Minimization	$x^{*2} = [-2.904, -2.904]$	$f = -78.3323$	2
Test Function 2	CTF	Cycle1	Minimization	$x^{*1} = [0.999, 0.999, 0.997, 0.994]$	$f = 2.1101e-4$	53
			Tunneling	$x^{01} = [0.999, 0.999, 0.997, 0.994]$	$f = 2.1101e-4$	36
		Cycle 2	Minimization	$x^{*2} = [1,1,1,1]$	$f = 0$	19
	ETF	Cycle1	Minimization	$x^{*1} = [0.999, 0.999, 0.997, 0.994]$	$f = 2.1101e-4$	53
			Tunneling	$x^{01} = [1,1,1,1]$	$f = 0$	18
		Cycle 2	Minimization	$x^{*2} = [1,1,1,1]$	$f = 0$	1
Test Function 3	CTF	Cycle1	Minimization	$x^{*1} = [0.972, 1, 1, 1, 1, 1, 1, 1]$	$f = 0.0306$	2
			Tunneling	$x^{01} = [0.972, 1, 1, 1, 1, 1, 1, 1]$	$f = 0.0306$	12
		Cycle 2	Minimization	$x^{*2} = [1, 1, \ldots, 1]$	$f = 5.8895e-32$	3
	ETF	Cycle1	Minimization	$x^{*1} = [0.972, 1, 1, 1, 1, 1, 1, 1]$	$f = 0.0306$	2
			Tunneling	$x^{01} = [0.978, 1.018, 0.994, 1.006, 0.934, 1.017, 1.01, 0.997]$	$f = 0.0210$	22
		Cycle 2	Minimization	$x^{*2} = [1, 1, \ldots, 1]$	$f = 5.8895e-32$	19

6 Conclusion

This paper considered the tunneling algorithm for finding the global minimizer of the nonlinear unconstrained minimization problem on $f(x)$ with continuous first and second derivatives. In this paper the performance of the CTF in comparison with the ETF in the tunneling phase of the tunneling algorithm was considered. In order to facilitate a comparison of the performance of the tunneling algorithm when using both CTF and ETF in the tunneling phase, three test functions were taken as objective functions. In conclusion the tunneling algorithm can locate the global minimizer of three test functions. The tunneling algorithm was verified by numerical experiments with typical 2-dimensional, 4-dimensional and 8-dimensional non-constrained global optimization problems. The results of this work support the idea that, the ETF is more efficient than the CTF because it can create a pole at a local minimizer independent of the accuracy of the local minimum found. The ETF found the tunneling point x^{01} in less iterations than CTF for test functions one and two. For all test functions ETF found the new tunneling point x^{01} which is different from previous local minimum x^{*1}.

Acknowledgements

I would like to thank to Universiti Teknologi Malaysia (UTM) for giving me the opportunity to do this research. I wish to express my deepest gratitude to my supervisor, Assoc. Prof. Dr. Rohanin Ahmad for all her encouragements and guidance.

References

[1] Levy, A. V. and Montalvo, A. The tunneling algorithm for the global minimization of Functions. *SIAM Journal on Scientific and Statistical Computing*. 1985. 6:15-29.

[2] Yuanqiao,W., Shengsheng, Y., Jingli, Z. and Liwen, H. Agent based distributed parallel tunneling algorithms. *PDCAT, LNCS 3320*. 2004. 226-229.

[3] Levy, A.V. and Gomez, S. The tunneling method applied to global optimization. *Numerical Optimization*, *SIAM*. 1985. 213- 244.

[4] Gomez, S. and Romero, D. Two global methods for molecular geometry optimization. *In Progress in mathematics*. 1994. 121: 503-509.

[5] Gomez, S., Del Castillo, N., Castellanos, L. and Solano, J. The parallel tunneling method. *Parallel Computing*. 2003. 29: 523-533.

[6] Yao, Y. Dynamic tunneling algorithm for global optimization. *IEEE Trans. Syst., Man, Cybern.*1989.19: 1222–1230.

[7] Molga, M. and Smutnicki, C. *3 kwietnia.* 2005:1-43.