J Glob Optim (2007) 38:61–77
DOI 10.1007/s10898-006-9083-3

ORIGINAL ARTICLE

A novel three-phase trajectory informed search methodology for global optimization

Jaewook Lee

Received: 29 June 2006 / Accepted: 7 August 2006 / Published online: 6 September 2006

Abstract A new deterministic method for solving a global optimization problem is proposed. The proposed method consists of three phases. The first phase is a typical local search to compute a local minimum. The second phase employs a discrete sup-local search to locate a so-called sup-local minimum taking the lowest objective value among the neighboring local minima. The third phase is an attractor-based global search to locate a new point of next descent with a lower objective value. The simulation results through well-known global optimization problems are shown to demonstrate the efficiency of the proposed method.

Keywords Global optimization · Dynamical systems · Deterministic methods

1 Introduction

Computation of the global optimal solutions for non-linear programming is important in a very broad range of applications. As its importance alone may suggest, global optimization is generally an extremely hard problem. From the complexity point of view, global optimization problems belong to the class of NP-hard problems ([30]), which means that the computational time required to solve the problem is expected to grow exponentially as the input size of the problem increases.

Despite such difficulties, however, numerous algorithms for solving global optimization problems have been developed during the last two decades due to increasing needs of most engineers and scientists to solve complex real-world systems. Most existing methods can be classified into two categories as in [12, 17]: stochastic methods and deterministic methods. Stochastic methods sample the objective function for a small number of points and include genetic algorithms, simulated annealing, clustering

J. Lee (✉)
Department of Industrial Engineering, Pohang University of Science and Technology, Pohang, Kyungbuk 790-784, Korea
e-mail: jaewookl@postech.ac.kr

methods, bayesian methods, etc. This class of methods does not need derivative information, but the required number of samples to arrive at a desired solution is often not admissible for large problems. On the other hand, deterministic methods typically provide a mathematical guarantee for convergence to an ϵ-global minimum in a finite number of steps for optimization problems involving certain mathematical structures. Branch and bound methods, cutting planes, decomposition-based approaches, covering methods, interval methods, interior point methods, and tunnelling methods are all deterministic in nature.

In this paper, a new deterministic methodology for global optimization is presented. This method consists of three-phases: Phase I for approaching a new local minimum in terms of a gradient system-based local search, Phase II for locating a sup-local minimum in terms of a discrete local search, and Phase III for approaching a feasible solution with a lower objective value than those of previously obtained solutions in terms of an attractor-based global search.

The proposed method was stimulated by the study of neural optimization network ([8, 9, 29]) and recent progress in tunnelling methods ([2, 24, 31]), for example, by a code called TRUST proposed by Barhen and co-workers ([2, 7]). Their algorithm, employing a descent-tunnelling methodology characterized by non-Lipschitzian terminal repeller and 1-Dimensional (1-D) search, has made a significant improvement in solving a variety of global optimization problems. For 1-D problems, the TRUST algorithm is designed to sweep the whole search space and thus find the global minimum. However, a direct extension of the 1-D scheme to multi-dimensional global optimization problems does not guarantee that the global optimum will always be found since it cannot capture a trajectory that tunnels near the periphery of the basin of attraction of a global minimum if the surface gradients are weak.

Our proposed method overcomes the above weakness of the tunnelling strategy by employing two innovative ideas. First, it builds a so-called optionally scaled system, which enables an attractor-based trajectory search to capture a trajectory that traverses near a feasible region with lower objective values. Second, it introduces a novel concept of neighboring local minima adjacent to a local minimum based on dynamical characterization, which enables a discrete sup-local search to speed up the search process.

2 Basic idea

A general global optimization problem is defined as follows:

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{1}$$

where f: $D \subset \Re^n \rightarrow \Re$ is assumed to be smooth and the set D will be referred to as the set of feasible points (or search space). A point $\mathbf{x}^* \in D$ is called a local minimum if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in D$ with $\|\mathbf{x} - \mathbf{x}^*\| < \delta$ for some $\delta > 0$.

The proposed method consists of three-phases:

- Phase I is a continuous local search and looks for a local minimum.
- Phase II is a discrete sup-local search and looks for a sup-local minimum with the lowest objective value among the neighboring local minima.
- Phase III is an attractor-based global search and looks for a feasible solution with a lower objective value than those values obtained in Phase II.

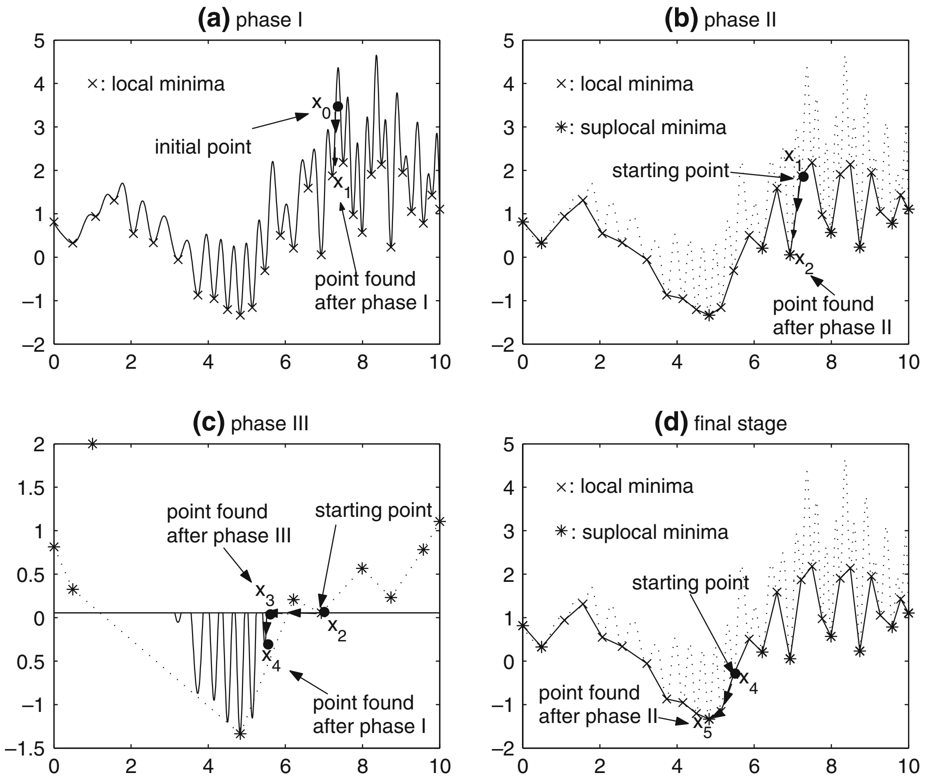

Fig. 1 Illustration of the proposed method for 1-D case

To illustrate the basic idea of the proposed method, consider a 1-D example shown in Fig. 1. Starting from a given initial point x_0, Phase I searches for a local minimum, say x_1, using a local search solver (see Fig. 1a). Next, Phase II iteratively generates a set of neighboring local minima starting from x_1, each of them is "locally" improved with respect to its neighboring local minima until it gets stuck at a locally optimal solution, say x_2, which we will call a sup-local minimum (see Fig. 1b). At a sup-local minimum x_2, Phase II is no longer helpful to find a new local optimal solution with lower objective value. Phase III is then invoked to escape from x_2 and to move on toward another point, say x_3, with lower objective value than $f(x_2)$. From x_3, Phase I is invoked to find a local minimum, say x_4, followed by Phase II, which locates a sup-local minimum, say x_5. In this way, the three phases are repeated alternatively in the search space until a global minimum is found or a stopping condition is met. In the next section, we describe a detailed procedure to handle a multi-dimensional global optimization problem where we specifically focus on the unconstrained case.

3 The proposed method

3.1 Phase I: local search phase

The aim of Phase I is to find a (first-order) local minimum. To this end, we build a generalized negative gradient system described by

$$\frac{d\mathbf{x}}{dt} = -\mathrm{grad}_R f(\mathbf{x}) \equiv -R(\mathbf{x})^{-1}\nabla f(\mathbf{x}), \tag{2}$$

where $R(\mathbf{x})$ is a positive definite symmetric matrix for all $\mathbf{x} \in \Re^n$. Such an R is called a *Riemannian metric* on $\Re^n$ [19]. It is interesting to note that many local search algorithms can be considered as a discretized implementation of process (2) depending on the choice of $R(x)$. For example, if $R(x) = I$, it is the naive steepest descent algorithm; if $R(x) = B_f(x)$ where B_f is a positive definite matrix approximating the Hessian, $\nabla^2 f(x)$, then it is the Quasi–Newton method; if $R(x) = [\nabla^2 f(x) + \mu I]$, it is the Levenberg–Marquardt method, and so on [27]. Otherwise specified, we assume that f is twice differentiable to guarantee the existence of a unique solution (or trajectory) $\mathbf{x}(\cdot)$: $\Re \to \Re^n$ for each initial condition $\mathbf{x}(0)$. Note that it can be shown that the trajectory $\mathbf{x}(\cdot)$ is defined on all $t \in \Re$ for any initial condition $\mathbf{x}(0)$ under a suitable re-parametrization [14].

A state vector $\bar{\mathbf{x}}$ satisfying the equation $\nabla f(\bar{\mathbf{x}}) = 0$ is called an *equilibrium point* (or *critical point*) of system (2). We say that an equilibrium point $\bar{\mathbf{x}}$ of (2) is *hyperbolic* if the Hessian of f at $\bar{\mathbf{x}}$, denoted by $H_f(\bar{\mathbf{x}})$, has no zero eigenvalues (i.e., $H_f(\bar{\mathbf{x}})$ is positive definite). Note that all the eigenvalues of the Jacobian of a gradient system are real since they are symmetric matrices. A hyperbolic equilibrium point is called a (asymptotically) *stable* equilibrium point (or an *attractor*) if all the eigenvalues of its corresponding Jacobian are positive and an *unstable* equilibrium point (or a *repellor*) if all the eigenvalues of its corresponding Jacobian are negative. A hyperbolic equilibrium point $\bar{\mathbf{x}}$ is called an *index-k equilibrium point* if the Jacobian of the hyperbolic equilibrium point has exactly k negative eigenvalues. The *basin of attraction* of a stable equilibrium point $\mathbf{x}_s$ is similarly defined as

$$A(\mathbf{x}_s) := \left\{\mathbf{x}(0) \in \Re^n : \lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}_s\right\}.$$

From a topological point of view, the basin of attraction $A(\mathbf{x}_s)$ is an open and connected set.

One nice property of this formulation is that every local minimum of the optimization problem (1) corresponds to a (asymptotically) stable equilibrium point of system (2), i.e., $\bar{\mathbf{x}}$ is a stable equilibrium point of (2) if and only if $\bar{\mathbf{x}}$ is an isolated local minimum for (1) [19]. Hence, the task of finding the local minima of (1) can be achieved via locating corresponding stable equilibrium points of (2). Another nice property is that all the generalized gradient system have the equilibrium points at the same locations with the same type [19], i.e., if $R_1(\mathbf{x})$ and $R_2(\mathbf{x})$ are Riemannian metrics on $\Re^n$, then the locations and the types of the equilibrium points of $\mathrm{grad}_{R_1} f(\mathbf{x})$ and $\mathrm{grad}_{R_2} f(\mathbf{x})$ are the same. This convenient property allows us to design a computationally efficient algorithm to find stable equilibrium points of (2), and thus to find local optimal solutions of (1).

Phase I can be simply implemented by numerically integrating system (2) or by a steepest-descent method. However, since we are primarily concerned with finding a stable equilibrium point, which is a limit point of a trajectory, rather than the trajectory itself, we have the freedom to choose any robust local search algorithm if it can locate a local minimum nearby an initial guess [5]. For example, a hybrid local search algorithm such as trust region-based methods, BFGS methods, or SQP methods using second-order information [1, 11, 27] (or line search methods, simplex methods or their modifications when no derivative information is available [21, 26, 27]) can be employed for computational efficiency. Notice that from the 'locality' of search in

Phase I, to avoid jumping out of the current basin of attraction, it is important not to take too large steps initially for the line search employed by a local search method.

3.2 Phase II: discrete sup-local search phase

When we get stuck at a local minimum in Phase I, one important issue is how to escape from a local minimum and move on toward another neighboring local minimum. For a 1-D case, a neighboring local minimum adjacent to a local minimum, $\mathbf{x}_s$, can be simply defined as the local minima nearest to $\mathbf{x}_s$ in each direction of $\Re$. In multi-dimensional cases, we define a neighboring local minimum adjacent to a local minimum as follows.

Definition 1 Let $\mathbf{x}_s$ be a local minimum of (1). We say that a local minimum $\mathbf{y}_s$ is *adjacent* to $\mathbf{x}_s$ if there exists an index-one equilibrium point $\mathbf{x}_1$ such that the 1-D unstable manifold $W^u(\mathbf{x}_1)$[1] of $\mathbf{x}_1$ converges to both $\mathbf{x}_s$ and $\mathbf{y}_s$ with respect to system (2).

Note that such an index-one equilibrium point $\mathbf{x}_1$ is on $\partial A(\mathbf{x}_s) \cap \partial A(\mathbf{y}_s)$. (See Fig. 2.)

From a theoretical viewpoint, this definition enables us to build a graph $G = (V, E)$ describing the connections between the local minima with the following elements:

1. The vertices V of G are local minima $\mathbf{x}_s^1, \ldots, \mathbf{x}_s^p$ of (1), where p is the total number of local minima
2. The edge E of G can only connect two local minima adjacent to each other; $\mathbf{x}_s^i$ is connected with $\mathbf{x}_s^j$ if, and only if, $\mathbf{x}_s^i$ is adjacent to $\mathbf{x}_s^j$.

It can be shown that the graph G is connected under some mild conditions.

The original optimization problem (1) can now be transformed into the following combinatorial optimization problem

$$\min_{\mathbf{x}_s \in \mathcal{L}^1} f(\mathbf{x}_s), \tag{3}$$

where $\mathcal{L}^1$ is the set of all the local minima.

Utilizing the concept of a neighboring local minimum, Phase II uses a discrete local search strategy on $\mathcal{L}^1$ to solve problem (3). Specifically, we first construct a user-defined neighborhood $\mathcal{N}$ where the neighborhood $\mathcal{N}(\mathbf{x}_s) \subset \mathcal{L}^1$ of $\mathbf{x}_s \in \mathcal{L}^1$ is defined as the set of multiple neighboring local minima adjacent to (or near) $\mathbf{x}_s$. With respect to the neighborhood $\mathcal{N}$, we will call a point $\mathbf{x}^* \in \mathcal{L}^1$ a *sup-local minimum* if $f(\mathbf{x}^*) \leq f(\mathbf{y}_s)$ whenever $\mathbf{y}_s \in \mathcal{N}(\mathbf{x}_s)$. (It should be noted that the sup-local optimality depends on the neighborhood function that is used and the neighborhood should guarantee the connectedness to be solvable. See Fig. 3.) Then we perform a discrete local search strategy to locate a sup-local minimum.

There are two widely-used discrete local search strategies for solving problem (3): best-neighboring and hill-climbing, each involving a search in $\mathcal{N}(\mathbf{x}_k)$ of the current $\mathbf{x}_k$. In a best-neighboring strategy, the point leading to the maximum decrease in the objective function value is selected to update the current point. Therefore, all points in $\mathcal{N}(\mathbf{x}_k)$ need to be searched every time, leading to computation complexity of $O(m)$, where m is number of points in $\mathcal{N}(\mathbf{x}_k)$. In hill-climbing, the first point leading to a decrease in the objective function value is selected to update the current point. Most often, best-neighboring strategies are more computationally expensive than hill-climbing.

[1] An unstable manifold $W^u(\bar{\mathbf{x}})$ is defined as the set of all points such that $\lim_{t\to-\infty} \mathbf{x}(t) = \bar{\mathbf{x}}$.

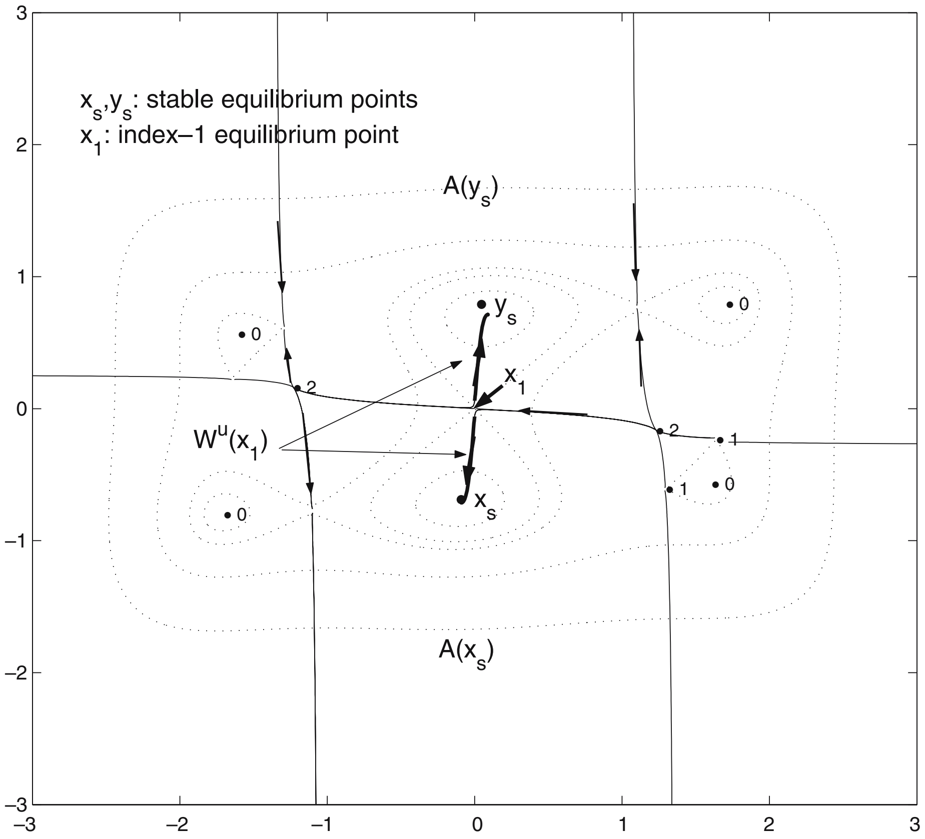

Fig. 2 A phase portrait of the negative gradient system corresponding to a six-hump camelback function (CA). The index-1 equilibrium point $\mathbf{x}_1$ is a type-one equilibrium point connecting two local minima $\mathbf{x}_\mathrm{s}$ and $\mathbf{y}_\mathrm{s}$

To locate a neighboring local minimum adjacent to a given local minimum, we may apply the following direct strategy: in order to escape from a local minimum, say $\mathbf{x}_\mathrm{s}$, we first move in reverse time from $\mathbf{x}_\mathrm{s}$ to an index-1 equilibrium point, say $\mathbf{x}_1$. Then, starting from $\mathbf{x}_1$, we advance time to approach an adjacent local minimum, say $\mathbf{y}_\mathrm{s}$ (see Fig. 2). The latter step can also be implemented using a local search algorithm as in Phase I. The former step, although it is hard to perform directly and time-consuming, can also be accomplished by adopting some of the techniques suggested in [19, 23]. This direct strategy can, however, be inefficient since we are only interested in locating an adjacent local minimum and do not need to find an exact index-1 equilibrium point, which is computationally intensive.

The following indirect strategy can instead be used to speed up the search of neighboring local minima: Choose an initial straight ray $\mathbf{r}(\lambda)$, emanated from $\mathbf{r}(0) = \mathbf{x}_\mathrm{s}$ and follow the ray $\mathbf{r}(\lambda)$, with increasing λ, (or employ a mixed cubic and quadratic interpolation line search algorithm) until it reaches its first local minimum (or it hits the boundary of search space) of $f(\mathbf{r}(\cdot))$. (The objective function f would increase, pass a local maximum, and reach a local minimum along the ray.) Starting from

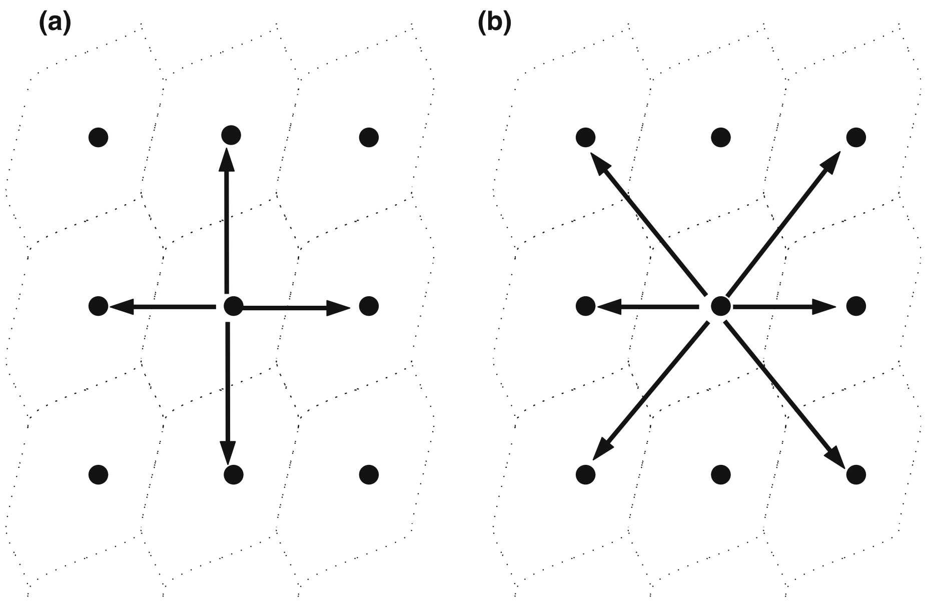

Fig. 3 User-defined neighborhood structure

the obtained point, locate a neighboring local minimum using a local search algorithm as in Phase I. (If a neighboring local minimum cannot be found, try another ray.) (see Fig. 3).

3.3 Phase III: attractor-based global search phase

When Phase II arrives at a sup-local minimum in $\mathcal{L}^2$, it stops and no longer proceeds to find a new feasible or local optimal solution with lower objective value than those of previously obtained solutions. One important issue is now how to escape from a sup-local minimum, say $\mathbf{x}^*$, and to find a new point of next descent in the lower level set given by

$$S(\mathbf{x}^*) = \left\{ \mathbf{x} \in \Re^n : f_{x^*}(\mathbf{x}) \overset{\text{def}}{=} f(\mathbf{x}) - f(\mathbf{x}^*) < 0 \right\}. \tag{4}$$

In general, the set $S(\mathbf{x}^*)$ can be decomposed into several disjoint connected components, i.e.,

$$S(\mathbf{x}^*) = \bigcup_{j=1}^{m} C_j,$$

where C_j are disjoint connected components and will be called a *feasible component* of (4).

Phase III aims at locating a point in a feasible component of (4) and is based on the following idea: if it is possible to find a nonlinear dynamical system such that all the feasible components are attractors (whose nearby trajectories approach it asymptotically) of the corresponding dynamical system, then it is possible to to find a feasible component of (4) by finding an attractor of the corresponding dynamical system.

To implement this idea, we build the following optionally scaled attracting dynamical system associated with f_{x^*}:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = -\rho(f_{x^*}(\mathbf{x}))\nabla f(\mathbf{x}), \tag{5}$$

where $\rho(\cdot)$ denotes a quasi-max zero function of one variable given by

$$\rho(s) = \begin{cases} 0, & \text{if } s \le 0, \\ m(s), & \text{if } s > 0 \end{cases}$$

and $m(s)$ is a strictly increasing differentiable function of $s \ge 0$ such that $m(0) = 0$ (hence, $m(s) > 0$ if $s > 0$).

Remark 1

(1) System (5) has a Lyapunov function given by $E(\mathbf{x}) := \frac{1}{2}\rho(f_{x^*}(\mathbf{x}))^2$ since

$$\frac{\mathrm{d}E(\mathbf{x})}{\mathrm{d}t} = -\rho(f_{x^*}(\mathbf{x}))^2 \|\nabla f(\mathbf{x})\|^2 \max\{m'(f_{x^*}(\mathbf{x})), 0\} \le 0$$

(2) In general, a quasi-max zero function $\rho(s)$ need not be differentiable at $s = 0$, but can be approximated by a differentiable function. One possible candidate for such a differentiable function is

$$\rho(s) = \frac{1}{\alpha}\ln(1 + \exp(\alpha m(s))) \simeq \max\{m(s), 0\},$$

where $\alpha > 1$ is a constant that affects the asymptotic behavior of the transformation. This choice of ρ combined with the assumption that f is twice differentiable guarantees the existence of a unique solution (or trajectory) $\mathbf{x}(\cdot)\colon \Re \to \Re^n$ for each initial condition $\mathbf{x}(0)$.

The next theorem establishes a result showing the relationship between feasible components of (4) and attractors of (5), which is a distinguished feature of system (5).

Theorem 1 *Every feasible component of* (4) *corresponds to an attractor of* (5), *i.e., if C is a feasible connected component of* (4)*, then C is an attractor of system* (5)*.*

Proof See Appendix. □

Theorem 1 asserts that the task of finding a new point in a feasible component of (4) can be achieved via locating its corresponding attractor of system (5). It should be, however, pointed out that the attractors of system (5) may not always be feasible component of (4). It is due to the fact that there is a possibility that the numerical integration of system (5) stops at a point $\bar{\mathbf{x}}$ where the Lyapunov function $E(\bar{\mathbf{x}}) > 0$.

Despite this fact, introducing system (5) endows the proposed method with a much wider convergent regions than the tunnelling algorithm suggested in ([2, 7]). To illustrate this, let us consider TRUST or its variants that employ a descent-tunnelling methodology characterized by non-Lipschitzian terminal repeller and 1-D search where the tunnelling is performed by integrating a dynamical system given by:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = (x_i - x_i^*)^{\frac{1}{3}}\theta(f_{x^*}(\mathbf{x})) \tag{6}$$

from the initial condition $x_i^* + \delta_i$ where $\theta(\cdot)$ is the Heaviside function, which is equal to 1 for positive values of the argument and zero otherwise. This system dynamics transforms a local minimum $\mathbf{x}^*$ into a terminal repeller $\mathbf{x}^*$ violating the Lipschitz condition and escapes this local minimum starting from any initial point nearby x^* in a finite time ([2, 7]). This method, however, fails to find a new region of next descent unless a system trajectory of (6) during tunnelling reaches a feasible component of (4), i.e., inside an attractor of system (5). Therefore, a large number of search directions are required to find a proper trajectory that tunnels the region, resulting in very high-computational complexity for higher dimensional cases. In contrast, the proposed method can capture a trajectory outside this attractor, or more exactly in the basin of attraction of this attractor, thereby, leading the captured trajectory to a feasible component of (4) (see Fig. 4).

3.4 Algorithm and implementation

We are now in a position to present a conceptual algorithm for the proposed method as follows.

Algorithm 1 Proposed Method

```
1. (Initialization)
   1.1. Initialize $\bar{\mathbf{x}}$ and a tolerance $\epsilon > 0$;
   1.2. Set $V = \emptyset$; // a set of local minima
2. (Phase I: Local Search Phase)
   2.1. Compute a local minimum $\mathbf{x}_s$ starting from $\bar{\mathbf{x}}$; set $V = V \cup \{\mathbf{x}_s\}$;
   2.2. if $\mathbf{x}_s$ satisfies the termination conditions, then stop; $\mathbf{x}_s$ is a desired solution;
3. (Phase II: Discrete Sup-Local Search Phase)
   do
   3.1. Compute a neighboring local minimum $\mathbf{y}_s \in \mathcal{N}(\mathbf{x}_s)$ of $\mathbf{x}_s$
   3.2. if $f(\mathbf{y}_s) < f(\mathbf{x}_s)$, then set $\mathbf{x}_s := \mathbf{y}_s$ and $V := V \cup \{\mathbf{x}_s\}$;
   until no neighboring local minimum is found that improves the current
   local minimum
   Set $\mathbf{x}^* := \mathbf{x}_s$; // $\mathbf{x}^*$ is a sup-local minimum //
4. (Phase III: Attractor-based Global Search Phase)
   for $i = 2$ to $l$ do
   4.1. choose a escaping point $\mathbf{x}_i$ from $\mathbf{x}^*$
   4.2. Locate an attractor of (5) starting from $\mathbf{x}_i$ (e.g., by numerically integration
        of (5));
        Set the obtained point by $\bar{\mathbf{x}}$;
   4.3. if $f(\bar{\mathbf{x}}) - f(\mathbf{x}^*) < \epsilon$, then goto Phase I;
   end
```

Remark 2

1. Any local search algorithm designed to locate a limit point of a trajectory of (5) starting from $\mathbf{x}_i$ can also be used in step 4.2 as discussed in Sect. 3.1.
2. The issue of how to choose an appropriate point $\mathbf{x}_i$ in step 4.1 of Phase III is not easy to address. This issue is raised since, as pointed out in the previous section, an attractor of system (5) may not always be a feasible component of (4). A simple heuristic strategy is to divide the search space into several rectangle regions and

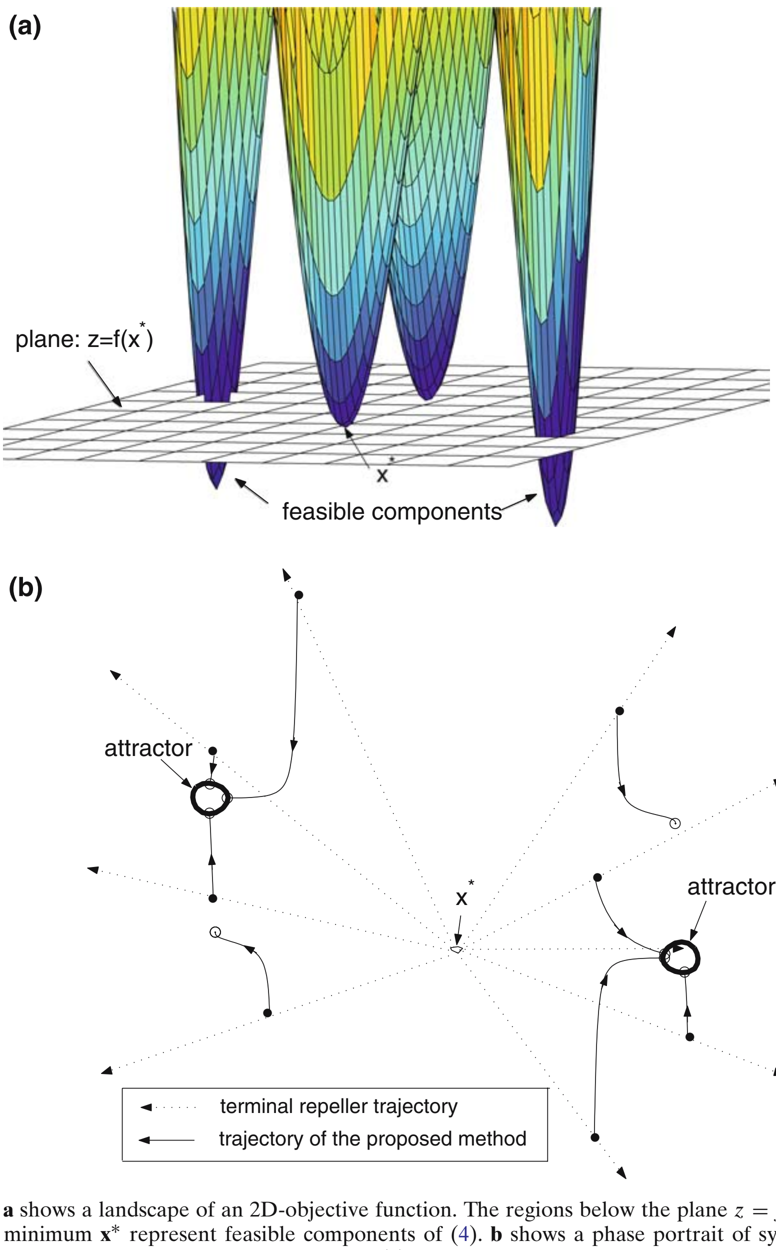
(a)
plane: z=f(x*)
x*
feasible components
(b)
attractor
attractor
x*
terminal repeller trajectory
trajectory of the proposed method

Fig. 4 **a** shows a landscape of an 2D-objective function. The regions below the plane $z = f(\mathbf{x}^*)$ for a local minimum $\mathbf{x}^*$ represent feasible components of (4). **b** shows a phase portrait of system (5) associated with f_{x^*}. Each feasible component in (a) is represented by an attractor of system (5). The marks '•' represent some escaping points generated by the repelling step and the *solid lines* represent the trajectories of system (5) starting from the points marked by '•'. These trajectories are attracted by the attractors of system (5), most of which converge to the feasible components of (4). The *dotted lines* represent the terminal repelled trajectory of system (6), most of which bypass feasible components of (4) to choose a random point in a region that does not contain previously obtained local minima listed in V. Another strategy, we will call a *double repelling step*, is as follows: Choose an initial straight ray $\mathbf{r}(\lambda)$, emanated from $\mathbf{r}(0) = \mathbf{x}_s$ and follow the ray $\mathbf{r}(\lambda)$, with increasing λ, (or employ a mixed cubic and quadratic interpolation line search algorithm) until it reaches its 2nd (or ith, $i \geq 2$) local minimum (or it hits the boundary of search space) of $f(\mathbf{r}(\cdot))$. (The objective function f would increase, pass a first local maximum, and reach a first local minimum along the ray, then pass a second local maximum, and reach a second local minimum and so on.)

4 Simulation results and discussion

4.1 Simulation Benchmark problems

The algorithm described in the previous section has been tested on several benchmark problems we have found in the literature ([2, 6, 7, 16]). The descriptions of the test examples are given in Appendix-B.

Tables 1 and 2 show the performance of our proposed algorithms compared to other global optimization methods for low-dimensional problems and high-dimensional problems, respectively. The following abbreviations are also used: PRS is the pure random search; MS the multistart; SA the simulated annealing method of ([22]); EA denotes the evolution algorithms of ([32]); GA refers the genetic algorithms of ([4, 16, 25]); MLSL is the multiple level single linkage method of ([20]); IA is the interval arithmetic technique of ([28]); TS refers to the tabu search methods of ([3, 6, 10]); MCS is the multilevel coordinate search method of ([18]); TN refers to the tunneling methods such as the TRUST algorithm in ([2, 7]). The criterion for comparison is the number of function evaluations. The results of some compared algorithms (PRS, MS, EA, MLSL, IA, TN in Table 1) are taken from ([2, 7]). The simulation results demonstrate that the proposed method works successfully in producing high-quality solutions (some of them are benefited from employed hybrid local search algorithms in Phase I) and is competitive with these state-of-art methods.

Table 1 Number of function evaluations required by different methods to reach a global minimum of low-dimensional test functions

	CA	GP	SH	BR	H3,4
PRS	–	5,125	6,700	4,850	5,280
MS	–	4,400	–	1,600	2,500
EA	–	460	–	430	–
MLSL	–	148	–	206	197
IA	326	–	7,424	1,354	–
GA	176	191	742	173	201
SA	1,903	–	780	505	1,459
TS	246	230	274	212	438
MCS	–	81	141	45	224
TN	31	103	72	55	58
Proposed	38	199	67	26	22

Table 2 Number of function evaluations required by different methods to reach a global minimum of high-dimensional test functions

	$S_{4,5}$	$S_{4,7}$	$S_{4,10}$	$H_{6,4}$	RT_{10}	R_{10}	RT_{20}	R_{20}	DP_{25}	L_{30}
GA	1,086	1,087	1,068	973	1,2078	6,340	–	–	–	–
TS	841	833	840	–	5,107	7,062	24,485	34,139	39,374	65,367
SA	–	–	–	–	1,08,243	1,05,137	81,454	74,512	–	1,06,798
MCS	298	178	384	146	1,963	1,233	5,957	4,247	30,376	1,661
TN	298	533	469	963	–	–	–	–	–	–
Proposed	443	480	328	65	343	723	1,045	1,743	1,863	324

4.2 Large-scale simulation

To evaluate the performance of the proposed method for a large-scale practical application, we have performed a simulation with a sonar data set. This is the data set used by Gorman and Sejnowski [13] in their study of the classification of sonar signals using a neural network. The data consists of sonar returns collected from two sources: a metal cylinder and a similarly shaped rock. Both objects were lying on a sand surface, and the sonar chirp projected at them from different angles produced the variation in the data. The data contains 111 patterns obtained from the metal cylinder, and 97 patterns obtained from the rock. Each pattern is a set of 60 continuous numbers in the range 0.0 – 1.0. For the simulation, a three layer network of size 60-5-1 is considered. This one consists of an input layer, a hidden layer, and an output layer, interconnected by modifiable weights, represented by links between layers. For each input $\mathbf{x}(n) \in \Re^{60}$, $n = 1, \ldots, 208$, the output of this neural network model is given by

$$z_k(n) = \phi\left(\sum_{j=1}^{5} w_{j+1}\phi\left(\sum_{i=1}^{60} w_{5i+j+6}x_i(n) + w_{j+6}\right) + w_1\right),$$

where $\mathbf{w} = (w_1, \ldots, w_{311})^T$ is a synaptic weight vector and ϕ: $\Re \to \Re$ is a nonlinear activation function (e.g. $\phi(s) = 1/(1 + \exp(-ay))$ for a positive gain a). The supervised training of a neural network is then viewed as an unconstrained minimization problem as follows:

$$\min_{\mathbf{w}} f(\mathbf{w}),$$

where the objective function f: $\Re^{311} \to \Re$ is a highly nonlinear mean squared training error (MSE) function.

Figure 5 shows the performance of our proposed algorithm compared to error back-propagation algorithm (EBP) [15], TRUST algorithm or its variant (TR), genetic algorithm (GA) and simulated annealing algorithm (SA). The result demonstrates a significant performance improvement of the proposed method compared with other algorithms.

5 Concluding remarks

In this paper, a new deterministic method for global optimization has been developed. The proposed method consists of three-phases: Phase I is a typical continuous local

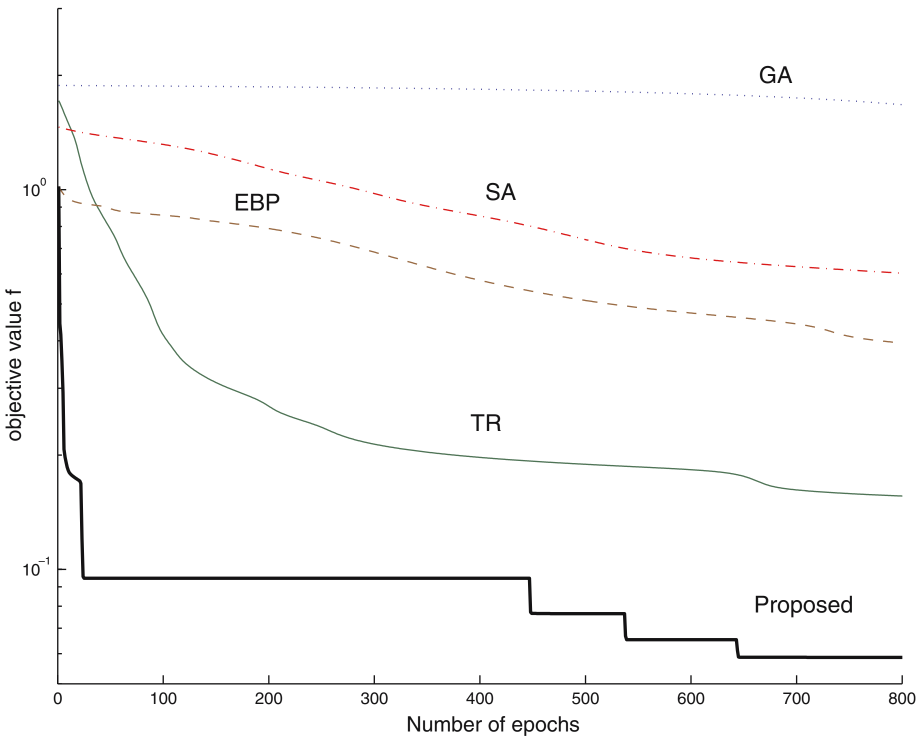

Fig. 5 Convergence curve for the sonar problem

search and looks for a local minimum. Phase II is a discrete sup-local search and looks for a sup-local minimum with the lowest objective value among the neighboring local minima. Phase III is an attractor-based global search and looks for a feasible solution with a lower objective value than those values obtained in Phase II.

The method has several features: first it does not require a good initial guess (although such a guess would of course reduce the number of steps needed). Second, it is better than stochastic methods such as multi-start methods since it can avoid many unnecessary efforts in re-determining already known minima. Third, it is very general and could be extended to any global optimization problem. An application of the method to more large-scale real-world problems remains to be investigated.

Appendix

Proof of Theorem 1

Proof Let C be a feasible connected component of the set $S(\mathbf{x}^*)$ and $\mathbf{x} \in \bar{C}$ where $\bar{C}$ is the closure of C. Since

$$f_{x^*}(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*) \leq 0 \quad \text{or} \quad \rho(f_{x^*}(\mathbf{x})) = 0$$

the Jacobian of $F(\mathbf{x}) \equiv -\rho(f_{x^*}(\mathbf{x}))\nabla f(\mathbf{x})$ at $\mathbf{x}$ is

$$\begin{aligned} J_F(\mathbf{x}) &= -\rho(f_{x^*}(\mathbf{x}))\nabla^2 f(\mathbf{x}) - \nabla f(\mathbf{x})\nabla f(\mathbf{x})^T \rho'(f_{x^*}(\mathbf{x})) \\ &= -\nabla f(\mathbf{x})\nabla f(\mathbf{x})^T \rho'(f_{x^*}(\mathbf{x})). \end{aligned}$$

If $\mathbf{x} \in C$, then $\rho'(f_{x^*}(\mathbf{x})) = 0$ and so we have $J_F(\mathbf{x}) = 0$. Next if $\mathbf{x} \in \partial C$, then $\rho'(f_{x^*}(\mathbf{x})) > 0$. Here we assume that ∂C is an $(n-1)$-dimensional manifold, which is a generic property by Morse–Sard Theorem. To show that the set $\bar{C}$ is an attractor of (5), we will show that $v^T J_F(\mathbf{x})v = 0$ for all $v \in T_x(C)$ and $v^T J_F(\mathbf{x})v < 0$ for all $v \in T_x(C)^\perp$ where $T_x(C)$ is an $(n-1)$-dimensional tangent space of C at $\mathbf{x}$ and can be described by

$$T_x(C) = \nabla f(\mathbf{x})^\perp = \{v : \nabla f(\mathbf{x})^T v = 0\}.$$

(1) For any vector $v \in \nabla f(\mathbf{x})^\perp = \{v : \nabla f(\mathbf{x})^T v = 0\}$, we have $v^T J_F(\mathbf{x})v = 0$.
(2) For any vector $v \notin \nabla f(\mathbf{x})^\perp$, i.e., $v = c\nabla f(\mathbf{x})$ where c is a scalar, we have

$$v^T J_F(\mathbf{x})v = -\rho'(f_{x^*}(\mathbf{x}))\|\nabla f(\mathbf{x})^T v\|^2 < 0$$

Therefore, C is an attractor of system (5). □

Test functions

(CA) or (HM) Hump Function (or Six-hump Camelback 2-D Function
Definition: $HM(\mathbf{x}) = 1.0316285 + 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
Search space: $-5 \le x_i \le 5, \quad i = 1, 2$
Global minimum: $\mathbf{x}^* = (0.0898, -0.7126), (-0.0898, 0.7126); HM(\mathbf{x}^*) = 0$

(GP) Goldstein and Price Function
Definition: $GP(\mathbf{x}) = \big(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\big)\big(30 + ((2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)\big)$
Search space: $-2 \le x_i \le 2, \quad i = 1, 2$
Global minimum: $\mathbf{x}^* = (0, -1); GP(\mathbf{x}^*) = 3$

(SH) Shubert Function
Definition: $SH(\mathbf{x}) = \Big(\sum_{i=1}^5 i\cos\big((i+1)x_1 + i\big)\Big)\Big(\sum_{i=1}^5 i\cos\big((i+1)x_2 + i\big)\Big)$
Search space: $-10 \le x_i \le 10, \quad i = 1, 2$
Global minimum: 18 global minima and $SH(\mathbf{x}^*) = -186.7309$

(BR) Branin RCOS Function
Definition: $RC(\mathbf{x}) = (x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$
Search space: $-5 \le x_1 \le 10, 0 \le x_2 \le 15$
Global minimum: $\mathbf{x}^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475); RC(\mathbf{x}^*) = 0.397887$

($H_{3,4}$) Hartmann Function
Definition: $H_{3,4}(\mathbf{x}) = -\sum_{i=1}^4 \alpha_i \exp\big[\sum_{j=1}^3 A_{ij}(x_j - P_{ij})^2\big], \alpha = [1, 1.2, 3, 3.2]^T$

$$A = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, P = 10^{-4}\begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$$

Search space: $0 \le x_i \le 1, \quad i = 1, 2, 3$
Global minimum: $\mathbf{x}^* = (0.114614, 0.555649, 0.852547); H_{3,4}(\mathbf{x}^*) = -3.86278$

($H_{6,4}$) Hartmann Function
Definition: $H_{6,4}(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left[\sum_{j=1}^{6} B_{ij}(x_j - Q_{ij})^2\right], \alpha = [1, 1.2, 3, 3.2]^T$

$$B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix},$$

$$Q = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}.$$

Search space: $0 \le x_i \le 1, \quad i = 1, \ldots, 6$
Global minimum: $\mathbf{x}^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300)$; $H_{3,4}(\mathbf{x}^*) = -3.32237$

($S_{4,m}$) **Shekel Function**
Definition: $S_{4,m}(\mathbf{x}) = -\sum_{j=1}^{m}\left[\sum_{i=1}^{4}(x_i - C_{ij})^2 + \beta_j\right]^{-1}, \beta = \frac{1}{10}[1, 2, 2, 4, 4, 6, 3, 7, 5, 5]^T$

$$C = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix}$$

Search space: $0 \le x_i \le 10, \quad i = 1, \ldots, 4$
Global minimum: $\mathbf{x}^* = (4, 4, 4, 4); S_{4,5}(\mathbf{x}^*) = -10.1532, S_{4,7}(\mathbf{x}^*) = -10.4029, S_{4,10}(\mathbf{x}^*) = -10.5364$

(RT_n) Rastrigin Function
Definition: $RT_n(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$
Search space: $-2.56 \le x_i \le 5.12, \quad i = 1, \ldots, n$
Global minimum: $\mathbf{x}^* = (0, \ldots, 0); RT_n(\mathbf{x}^*) = 0$

(R_n) Rosenbrock Function
Definition: $R_n(\mathbf{x}) = \sum_{i=1}^{n-1}\left[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right]$
Search space: $-5 \le x_i \le 10, \quad i = 1, 2, \ldots, n$
Global minimum: $\mathbf{x}^* = (1, \ldots, 1); R_n(\mathbf{x}^*) = 0$

(DP_n) Dixon & Price Function
Definition: $DP_n(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
Search space: $-10 \le x_i \le 10, \quad i = 1, \ldots, n$
Global minimum: $\mathbf{x_i^*} = 2^{-\left(\frac{2^i - 2}{2^i}\right)}, \quad i = 1, \ldots, n; DP_n(\mathbf{x}^*) = 0$

(L_n) Levy Function
Definition: $L_n(\mathbf{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1}\left[(y_i - 1)^2\left(1 + 10\sin^2(\pi y_i + 1)\right)\right] + (y_n - 1)^2\big(1 +$

$10\sin^2(2\pi y_n)\big)$
$y_i = 1 + \frac{x_i - 1}{4}, i = 1, \ldots, n$
Search space: $-10 \leq x_i \leq 10, \quad i = 1, \ldots, n$
Global minimum: $\mathbf{x}^* = (1, \ldots, 1); L_n(\mathbf{x}^*) = 0$

Acknowledgements This work was supported by the KOSEF under the grant number R01-2005-000-10746-0.

References

1. Al Baali, M.: Improved Hessian approximation for the limited memory BFGS method. Numer. Algorithms, **22**, 99–112 (1999)
2. Barhen, J., Protopopescu, V., Reister, D.: TRUST: a deterministic algorithm for global optimization. Science, **276**, 1094–1097 (1997)
3. Battiti, R., Tecchiolli, G.: The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. Ann. Oper. Res. **63**, 153–188 (1996)
4. Bessaou, M., Siarry, P.: A genetic algorithm with real-value coding to optimize multimodal continuous functions. Struct. Multidisc. Optim. **23**, 63–74 (2001)
5. Brown, A.A., Bartholomew-Biggs, M.C.: Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations. J. Optim. Theory Appl. **62**, 211–224 (1989)
6. Chelouah, R., Siarry, P.: Tabu search applied to global optimization. Eur. J. Oper. Res. **123**, 256–270 (2000)
7. Cetin, B.C., Barhen, J., Burdick, J.W.: Terminal repeller unconstrained subenergy tunnelling. J. Optim. Theory Appl. **77**(1), 97–126 (1993)
8. Chua, L.O., Lin, G.N.: Nonlinear optimization with constraints. IEEE Trans. Circ. Syst. **CAS-31**, 182–188 (1984)
9. Chua, L.O.: Global optimization: a naive approach. IEEE Trans. Circ. Syst. **37**, 966–969 (1990)
10. Cvijovic, D., Klinowski, J.: Taboo search: An approach to the multiple minima problem. Science **267**, 664 (1995)
11. Di Fiore, C., Fanelli, S., Lepore, F., Zellini, P.: Matrix algebras in Quasi-Newton methods for unconstrained minimization. Numer. Math. **94**, 479–500 (2003)
12. Floudas, C.A., Pardalos, P.M.: Recent advances in Global Optimization. Princeton series in Computer science, NJ (1992)
13. Gorman, R.P., Sejnowski, T.J.: Analysis of hidden units in a layered network trained to classify sonar targets. Neural Netw. **1**, 75–89 (1988)
14. Guckenheimer, J., Homes, P.: Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields. Applied math. sci. vol. 42, Springer, New York (1986)
15. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, New York (1999)
16. Hedar, A., Fukushima, M.: Minimizing multimodal functions by simplex coding genetic algorithm. Optim. Methods Softw. **18**, 265–282 (2003)
17. Horst, R., Pardalos, P.M.: *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht (1994)
18. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search. J. Global Optim. **14**, 331–355 (1999)
19. Jongen, H.Th., Jonker, P., Twilt, F.: *Nonlinear Optimization in* $\Re^n$, Peter Lang Verlag, Frankfurt (1986)
20. Kan, A.H.G.R., Timmer, G.T.: A stochastic approach to global optimization. In: Boggs, P.T., Byrd, R.H., Schnabel, R.B. (eds.), Numerical Optimization pp. 245–262. SIAM, Philadelphia, PA (1985)
21. Kelley, C.T.: Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. SIAM J. Optim. **10**, 43–55 (1999)
22. Kirkpatrick, S., Gelatt, C.D., Timer, G.T.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
23. Lee, J.: Dynamic gradient approaches to compute the closest unstable equilibrium point for stability region estimate and their computational limitations. IEEE Trans. Autom. Control **48**(2), 321–324 (2003)
24. Levy, A.V., Montalvo, A.: The tunnelling algorithm for the global minimization of functions. SIAM J. Sci. Stati. Comput. **6**, 15–29 (1985)

25. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin (1996)
26. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
27. Nocedal, J.N., Wright, S.J.: *Numerical Optimization*. Springer-verlag, New York, Heidelberg, Berlin (1999)
28. Ratschek, H., Rokne, J.: *New Computer Methods for Global Optimization*. Ellis Horwood, Chichester, UK (1988)
29. Tank, D.W., Hopfield, J.J.: Simple 'neural' optimization networks: an A/D converter, signal decision circuit, and a linear porgramming circuit. IEEE Trans. Circ. Sys. **CAS-33**(5), 533–541 (1986)
30. Vavasis, S.: *Nonlinear Optimization: Complexity Issues*. Springer-Verlag, New York (1990)
31. Yao, Y.: Dynamic tunnelling algorithm for global optimization. IEEE Trans. Sys. Man Cybern. **19**, 1222–1230 (1989)
32. Yong, L., Lishan, K., Evans, D.J.: The annealing evolution algorithm as function optimizer. Parallel Comput. **21**, 389 (1995)