Journal of Optimization Theory and Applications (2019) 180:855–878
https://doi.org/10.1007/s10957-018-1397-z

Gradient Surfing: A New Deterministic Approach for Low-Dimensional Global Optimization

Efrat Taig[1] · **Ohad Ben-Shahar**[2]

Received: 1 June 2018 / Accepted: 17 September 2018 / Published online: 27 September 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

We describe a novel global optimization technique which utilizes global minima basins of attraction in order to quickly converge to a global minima. A key to the proposed method is the "steeper goes deeper" heuristic: coupling between magnitudes of gradients on different level sets of a basin of attraction and the depth of its minima. Local minima are avoided with a combination of local optimization and a heuristic-based leaping step. Gradient surfing performance is evaluated across a set of small-scale problems from the literature, and results are compared to those of 12 previously published methods. A practical six-dimensional non-convex image registration application is presented as well, where GS performance exceeds that of classic global optimization methods in both speed and accuracy. Additionally, we validate the optimization method by applying a new Gaussian mixture model benchmark for non-convex function. Finally, the "steeper goes deeper" heuristic is validated empirically on five different classes of non-convex functions using two different evaluation approaches. In all cases, steeper gradients are shown to lead to deeper optima with a high probability.

Keywords Global optimization · Local optimization · Basin of attraction · Level set · Gradient descent

Communicated by Francesco Zirilli.

✉ Efrat Taig
dubie@post.bgu.ac.il

Ohad Ben-Shahar
ben-shahar@cs.bgu.ac.il

[1] Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva, Israel

[2] Department of Computer Science, Ben-Gurion University of the Negev, Beer Sheva, Israel

1 Introduction

Optimization, the process of obtaining the minimum of some general cost function (also termed loss, objective, fitness, or energy function), is a key to many scientific and technological developments, tools, and applications: from machine learning (e.g., optimization of model parameters [1]) and computer vision (e.g., image transformation and curve completion [2–4]), to economics (e.g., minimize cost, maximize revenue [5]), chemistry (e.g., chemical reaction paths [6,7]), physical sciences (e.g., energy minimization [8,9]), geophysics (e.g., seismic signals estimation [10]), biology (e.g., protein folding [11]), industrial engineering (e.g., warehouse maintenance, optimal transportation [12]) and many more. The need to find the best set of parameters that optimize the performance of a system is in many cases an essential ingredient of the computation.

With the consistent improvement in computational power, global optimization research has seen significant advancement in recent decades while an increasing number of approaches have been proposed [13–15]. Among the various approaches, we refer in particular to a set of methods that share common ground with ours. The *Tunneling*-based method, first introduced in [16] and further elaborated in [10,17–19], is a deterministic and iterative global optimization method, where each iteration consists of two phases: a local minimization phase that decreases the current value of the function until a local minimum is found, and a *tunneling* phase that seeks a point that differs from the current local minimum, such that when the new point is employed as starting point for the next minimization phase, the new obtained local minimum will have a function value no greater than the previous local minimum found. In a way, in our GS method, we used the tunneling idea of dividing each iteration into two phases: a local minimization phase and a global search phase. However, we significantly differ in our approach for dimensions higher than 1: While in the tunneling method the scanning has no explicit defined direction or procedure for propagation, thus having the same difficulty as the original minimization problem [20], in the GS method we apply the global search phase on the level sets of the objective function, a readily accessible structure, by using a well-defined level set scanning procedure.

Other relevant approaches that enabled explicit leaping between the different basins of attraction are the taboo [21] and simulated annealing [22] approaches, further elaborated in [23,24]. In taboo search, when no local move can offer improvement in the cost function, a leap to a random nearby neighboring point that exacerbates the cost is allowed as long as that point is not in some list of prohibited (taboo) points. In simulated annealing, such leaps to a neighboring point, selected by applying a small random change to the current point, are performed probabilistically and controlled by the annealing procedure.

Of special importance is the family of optimization tools based on the gradient descent (GD) approach whose simplicity and accessibility have made it one of the most common methods for optimization [20]. However, GD, in its simplest form, relies on the convexity of the function to be minimized, i.e., that the function has only one minimum or that the initial guess is always placed in the convex basin of attraction of the global minimum. This convexity condition, or the ability to make a "proper" initial guess, rarely applies to the systems of interest, thus requiring extended

schemes built on top of GD, either deterministic (e.g., trust [10], branch-and-bound methods [25], dynamic tunneling [17]) or stochastic (e.g., simulated annealing [26], taboo search [21]). In a recent publication, the gradient descent approach is utilized for a basin-of-attraction-based *Function Modification*, where in order to escape from the basin of attraction of a local minimum, a suitable Gaussian-based filling modified function is constructed [20].

While numerous global optimization methods are founded on the concept of sequentially improving the local optima found, in this paper we present a novel heuristic that can prevent existing descent methods to find a local optima in the first place. The GS method is a new deterministic global optimization method that can be easily embedded to a variety of frameworks and applications, with no additional knowledge on the cost function except the ability to evaluate (analytically or numerically) its value and its gradient at arbitrary points.

In this paper, the GS algorithm is applied to a set of small-scale problems from the literature [10,17,21,27] and the numerical results are reported alongside previously obtained results with 12 other methods for comparison. A new proposed non-convex benchmarks, together with a six-dimensional non-convex image registration application, are presented, and the GS results are compared to commonly used classical global optimization methods: simulated annealing (SA) [28], particle swarm optimization (PSO) [29,30], and genetic algorithm (GA) [31–33] (Sect. 4). Addition to the extensive empirical analysis presented in this paper, we believe that the novel "steeper goes deeper" heuristic introduced in this paper has the capacity to open a new path of research in the community and provide an alternative minimization procedure. In Sect. 2, the heuristic is exemplified and validated on 5 different general common classes of non-convex functions using 2 different evaluation approaches.

The Intuition Behind the GS Method The GS method combines two concepts—a standard gradient descent step that *locally* minimizes the function value, and a "leaping step" that changes the current search point to a different point in the search space where getting trapped in a local minimum is less likely.

Fundamental to our new proposal is a new leaping approach that is performed deterministically on the *level set* of the object function. The use of this readily accessible structure of the objective function, defined as the collection of points in the search space that map to the same value of the cost function, is best described by the following analogy (Fig. 1): Consider standing somewhere on a mountain range (red dot in Fig. 1), looking to find the lowest valley in a given area. Standard gradient descent method leads downhill by choosing the direction of progression as the direction of steepest descent at each point. This strategy might converge to a local valley; this strongly depends on the location of the starting point (see dashed red trajectory in Fig. 1). Our new method suggests that instead of descending from the current point, first "hike" the level set of the mountain and look for the point with the steepest slope (black dot on the white contour in Fig. 1), and only then continue to descend from that point. Assuming that the mountain range is smooth enough (i.e., has no abrupt cliffs), this hiking strategy has much better chances to penetrate deeper into the landscape and thus ending up in the lowest valley (see Sect. 3). This outcome is even more likely if the hiking of level sets is repeated at different (or even all) heights.

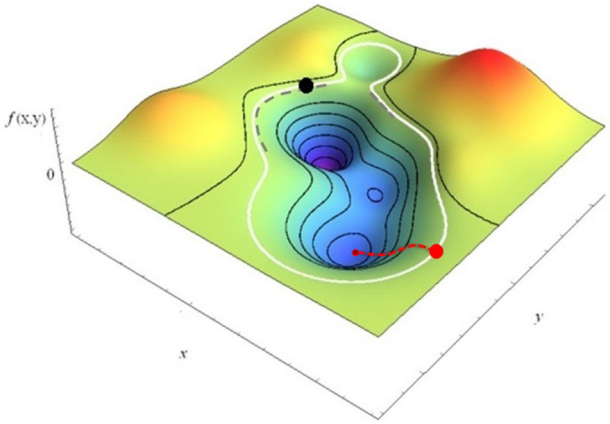


Fig. 1 Gradient surfing (GS) versus gradient descent (GD) on a non-convex function. If optimization begins from the red dot, GD will follow the dashed red line to a local minimum. However, first surfing along the dot's level set (colored in white), a point with even steeper descent (black point) will allow the same GD procedure to reach the global minimum. Note that GS will allow successful global optimization from *any* initial point along the white level set, while GD will only do so from a fraction of that level set (marked with dashed gray)

In energy function terms, scanning the level set has zero cost (since all points along the level set have equal energy value). Still, the extended optimization trajectory that seeks steepest descent direction among *all points along a level set* rather than at a single point (as in GD) is likely to yield a much better solution, perhaps even the *global* one, as is indeed the case in Fig. 1. Since leaping across the energy landscape and seeking better gradients is done without changing height, we dub the entire scheme *Gradient Surfing* (GS). In the rest of the paper, we first build the case for the underlying leaping heuristic. We then formally describe the GS algorithm in 2D, followed by proper approximations in arbitrary dimensions. We finish by demonstrating the power of the method using three benchmarks, one based on a set of standard test functions, one based on a set of Gaussian mixture models (GMM), and finally when applied to an image processing application.

2 The "Steeper Goes Deeper" Heuristic

As described informally above, the proposed GS method is based on seeking a point of greatest gradient magnitude among all points of the level set so the descent can continue from there and possibly escape local minima. Implicitly, then, this approach assumes that steeper gradients could easily lead to deeper points on the cost function terrain. In this section, we formalize this heuristic and explore the extend to which it is valid.

Recall that a *basin of attraction* (BOA) is the set of points in the search space such that any initial point $\mathbf{p}$ chosen in this set dynamically evolves with GD to a particular local minimum attractor, denoted by $GD(\mathbf{p})$. Each local minimum (and thus the global one also) has a unique BOA associated with it, and all these BOAs are disjoint. BOAs

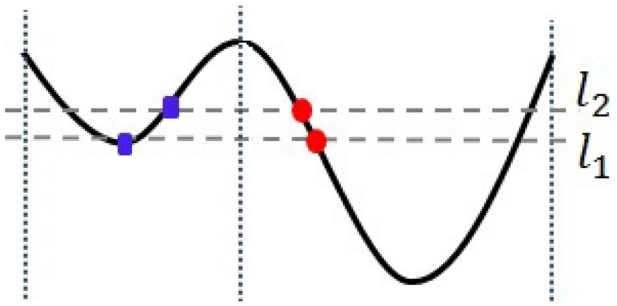


Fig. 2 The "steeper goes deeper" heuristic exemplified. This GMM is a 1D non-convex function with one global minima (that is part of the global BOA) and one local minima (in local BOA). Vertical dotted lines mark BOA's boundary and horizontal lines depict different level sets. On each level set l_i that is shared by both BOAs, red points belonging to the global BOA (red) have larger gradient than the corresponding blue points in the local BOA. If one is able to scan level sets to traverse BOAs, points with greater gradient may represent a more promising condition for local optimization procedure such as GD to converge to the global minimum

that contain global minima will be called *global BOAs*, while all others are *local BOAs*. With these notions in mind, our heuristic can be spelled out as follows:

> **The "steeper goes deeper" heuristic:** BOAs with deeper associated minima are more likely to have larger gradient magnitudes *at any given level set of the function*.

Note that this heuristic is *not* phrased in terms of the distribution of gradients in entire BOAs, and rather restricted to individual level sets of the objective function. The reason is twofold. First, "deeper," the goal of optimization, is a *relative* term and the level set provides an explicit reference "height" for comparison. Second, and more operationally, in many cases level sets allow to leap between BOAs.

The proposed "steeper goes deeper" heuristics clearly is true in certain common cases. Consider, for example, a Gaussian mixture models (GMM) of two 1D Gaussians with similar standard deviation but different amplitude as shown in Fig. 2. Clearly, the heuristics is valid for level set l_1 that goes through the local minimum, as the gradient vanishes at the local minimum but not for the corresponding points on l_1 that belong in the global BOA. For continuity arguments, the heuristics is valid for level sets close to l_1 and, in fact, it is valid for numerous cases even for such GMM with different standard deviation (see "Appendix A").

The example just discussed also illustrates why no function can violate the "steeper goes deeper" heuristic at *all* level sets simultaneously, but it is quite clear that an adversary can construct functions where it is invalid in a (perhaps even large) subset of level sets . The utility of this heuristic is thus highly related to how ubiquitous it is in practice, a question that is primarily empirical and possibly dependent on the domain of functions of interest. We thus set to explore this issue in various ways as described below (and also "Appendix A").

Empirically, testing the validity of our heuristic can be done in a rather explicit fashion, by exploring the relationship between the magnitude of gradient at a point **p**

and how deep one would get by locally optimizing with GD from that point. Let us denote these two functions by $||\nabla f(\mathbf{p})||$ and $\Psi(\mathbf{p})$, respectively, where the latter is defined as

$$\Psi(\mathbf{p}) = f(\mathbf{p}) - f(GD(\mathbf{p})),$$

where f is the objective function and $GD(\mathbf{p})$, as defined above, is the local minimum associated with the BOA of $\mathbf{p}$. If our heuristic is sound, the greater $||\nabla f(\mathbf{p})||$ is, so does $\Psi(\mathbf{p})$ should be (or vice versa), and thus testing this correlation for a large number of points for a large collection of objective functions of interest can validate or refute the utility of the heuristic. One way of testing this correlation is to collect numerous $\langle \Psi(\mathbf{p}), ||\nabla f(\mathbf{p})|| \rangle$ pairs from each level set of every test function, fit regression lines, and examine their slopes. Positive slopes would support the heuristic, while negative slopes would weaken it.

We performed the test above on several classes of non-convex functions. Perhaps, the most common one often used in science and engineering is general GMMs defined as the sum of arbitrary number of Gaussians in a predefined dimension. Here, we first explored the validity of the "steeper goes deeper" heuristic for 2D GMMs by creating a set of such GMMs functions, each being a sum of 10 Gaussians whose mean is a uniform random variable in the nominal domain $[-1, 1] \times [-1, 1]$, while their standard variation is another uniform random variable in the interval $\sigma \in [0.2, 0.3]$. Each of these GMMs was non-convex with an average number of 4 local minima in the domain, and to simplify exploration of level sets we normalized all of the obtained functions to $[0, 100]$. Testing for correlation of many $\langle \Psi(\mathbf{p}), ||\nabla f(\mathbf{p})|| \rangle$ pairs taken from 5 different level sets $\{20, 30, 40, 60, 80\}$ of such GMMs, we found that 81% of the regression lines exhibited positive slopes distributed compared to only 17% negative slopes, as shown in Fig. 3A.

Other classes of functions tested similar to the above-included higher dimension GMMs, trigonometric functions, and a mixture model of trigonometric functions and polynomials. While the details are in "Appendix A," in all cases we clearly find that the "steeper goes deeper" heuristic is significantly more often valid than not (see Fig. 3), paving the way for the type of global optimization algorithms discussed in the next section.

3 The Gradient Surfing Method

The description of the proposed GS method based on the "steeper goes deeper" heuristic is presented in the following sections, first for the two-dimensional case and then for higher dimensions.

3.1 Formulation: 2D Gradient Surfing

Before addressing optimization in arbitrarily large dimensions, we first limit ourselves to two-dimensional functions, where common benchmarks are available. Toward that

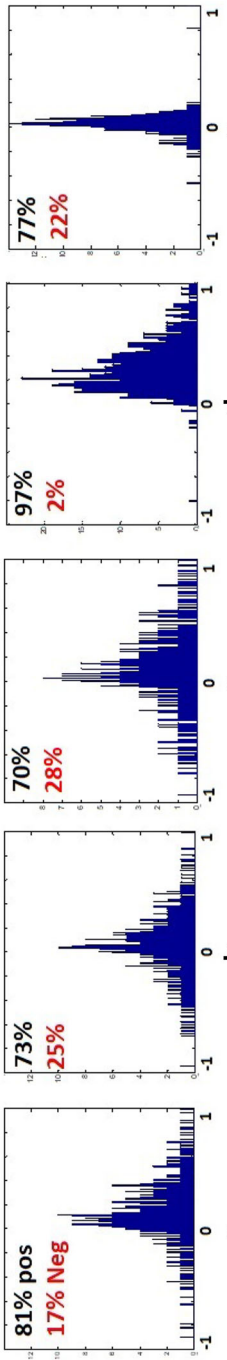


Fig. 3 Histogram of slopes of regression lines between $||\nabla f(\mathbf{p})||$ and $\Psi(\mathbf{p})$ for different non-convex functions (see also "Appendix A"). Insets show the percentage of positive (top) and negative (bottom, red) slopes that support the soundness of the "steeper goes deeper" heuristic. **a** Slopes distribution for 2D 10-GMM functions. **b** Slopes distribution for 3D 10-GMM functions. **c** Slopes distribution for 4D 10-GMM functions. **d** Slopes distribution for trigonometric functions. **e** Slopes distribution for a mixture of the trigonometric functions and fourth-order polynomials. See "Appendix A" for additional details about all functions

end, we define the unconstrained global optimization problem as follows:

$$\mathbf{P}^* = \underset{\mathbf{P}=(p_1,p_2)}{\operatorname{argmin}} \quad f(\mathbf{P}) : \Omega \subset \mathcal{IR}^2 \to \mathcal{IR}, \tag{1}$$

where f is the function to be optimized and Ω is the (typically compact) set of all feasible solutions. Since maximization of f is equivalent to minimizing $-f$, we limit our discussion to the problem of minimization. We now combine the gradient descent (GD) and level set surfing concepts into the gradient surfing (GS) iterative global optimization scheme, where each iteration consists of two steps:

(a) A *local minimization* step that decreases the current value of the function at iteration i such that $f(\mathbf{P_i}) < f(\mathbf{P}_{i-1})$. This is done by a standard gradient descent step

$$\mathbf{P_i} = \mathbf{P}_{i-1} + \alpha \cdot \nabla f(\mathbf{P}_{i-1}), \tag{2}$$

where α is the GD step size and $P_i, P_{i-1} \in \Omega \subset \mathcal{IR}^2$. GD can of course be substituted by more elaborate or higher-order descent steps.

(b) A *level set surfing* step that seeks a point $\mathbf{S}_i \in \Omega$ on the same level set of $\mathbf{P}_i$ (i.e., $f(\mathbf{S}_i) = f(\mathbf{P}_i)$) with the maximal gradient magnitude. To do so, the surfing starting point is initialized as

$$\mathbf{S}_i^0 = \mathbf{P}_i \tag{3}$$

and is updated according to:

$$\mathbf{S}_i^{j+1} = \mathbf{S}_i^j + \beta \cdot \mathbf{V}_i^j, \tag{4}$$

where β is the surfing step size and $\mathbf{V}_i^j$ is tangent to the level set curve (or perpendicular to the gradient, i.e., $\mathbf{V}_i^j \cdot \nabla f(\mathbf{S}_i^j) = 0$)

$$\mathbf{V}_i^j = \left(-\frac{\partial f(\mathbf{S}_i^j)}{\partial p_2}, \frac{\partial f(\mathbf{S}_i^j)}{\partial p_1} \right). \tag{5}$$

Since by construction we have $f(\mathbf{S}_i^{j+1}) = f(\mathbf{S}_i^j)$, the surfing essentially *scans* the level set and continues until satisfying a stopping criteria that can be a combination of domain-related terms (such as hitting the boundaries of the search domain Ω), time-related terms (e.g., number of function evaluations), or a circularity term (i.e., search is terminated once we return to the initial starting point). When the level set scanning is completed, we set

$$\mathbf{S}_i = \underset{S \in \left\{ \mathbf{S}_i^j : j=0,1,\ldots \right\}}{\operatorname{argmax}} \quad ||\nabla f(S)|| \tag{6}$$

and $\mathbf{S_i}$ then becomes the starting point for the next local minimization step.

The stopping criteria of the minimization step (and thus of the optimization as a whole) are the standard GD criteria. Algorithm 1 sketches in pseudo code the details of this

procedure while its stability, and the significance of the α and β constants are studied in Sect. 4.4.

Algorithm 1 $2D$ Surfing Method

Input: $f : \Omega \subset \mathcal{IR}^2 \rightarrow \mathcal{IR}$, $\mathbf{P_0}$
$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  $\mathbf{P_i} \leftarrow \mathbf{P}_{i-1} + \alpha \cdot \nabla f(\mathbf{P}_{i-1})$ % GD step.
  $\mathbf{S}_i^0 \leftarrow \mathbf{P}_i$
  $j \leftarrow 0$
  repeat % Surfing
    $\mathbf{S}_i^{j+1} \leftarrow \mathbf{S}_i^j + \beta \cdot \mathbf{V}_i^j$
    $j \leftarrow j + 1$
    $\mathbf{S_i} \leftarrow \underset{\mathbf{S} \in \left\{\mathbf{S}_i^j : j=0,1,\ldots\right\}}{\text{argmax}} ||\nabla f(\mathbf{S})||$
  until {Level set surfing termination conditions}
until {Termination conditions for GD method are satisfied}
$\mathbf{P}^* \leftarrow \mathbf{P_i}$

3.2 Formulation: Higher-Dimensional Gradient Surfing

The high-dimensional unconstrained global optimization problem is the extension of Eq. 1 to $\mathcal{IR}^n$, i.e.,

$$\mathbf{P}^* = \underset{\mathbf{P}=(p_1,p_2,\ldots p_n)}{\text{argmin}} \quad f(\mathbf{P}) : \Omega \subset \mathcal{IR}^n \rightarrow \mathcal{IR}. \tag{7}$$

While pursuing GS as a solution method, the extension of the minimization step from Sect. 3.1 to higher dimensions is the trivial extension of GD from Eq. 2, i.e.,

$$\mathbf{P_i} = \mathbf{P}_{i-1} + \alpha \cdot \nabla f(\mathbf{P}_{i-1}), \tag{8}$$

where this time $P_i, P_{i-1} \in \Omega \subset \mathcal{IR}^n$. However, the rest of the formulation from Sect. 2 cannot be trivially extended to higher dimensions since it is based on "surfing" along a level set. In higher dimensions, a level set is an N-dimensional manifold that is neither linearly nor cyclically ordered to facilitate its surfing, at least not easily so.[1] In what follows, we suggest two schemes for addressing this issue.

3.3 Iterative Parameters

One possible way to extend GS to multivariate cost functions f with $n > 2$ parameters is to iteratively fix $n-2$ parameters and apply the 2D surfing method for the projection

[1] One could, in theory, employed space filling curves [34] to scan higher-order manifolds systematically. Clearly, such an approach is not likely to produce a practical extension of our proposed optimization scheme to higher dimensions and thus we exclude it.

of f spanned by the two free parameters p_k and p_m. This step can be repeated until each parameter participates in at least one 2D projection, or more systematically until all pairs of parameters are selected. Algorithm 2 summarizes this optimization approach in pseudo code.

Algorithm 2 Iterative parameters

Input: $f : \mathcal{IR}^n \rightarrow \mathcal{IR}$, $\mathbf{P_0}$
repeat

- Choose 2 out of n parameters (p_k and p_m)
- Hold the other $n-2$ parameters fix
- Estimate and update p_k and p_m according to Algorithm 1

until {Termination conditions for GD method are satisfied}

3.4 Recursive Approach

As we attempt to expand the gradient surfing to higher dimensions, we could also leverage recursion in order to progressively reduce the dimensionality of the problem. Recall that as in the 2D case, we seek a point $\mathbf{S}_i \in \Omega$ on level set L_s that has the maximal gradient magnitude. This surfing operation can be formulated as a *constrained* optimization problem

$$\begin{aligned} \mathbf{S_i} = & \underset{\mathbf{P}\in\Omega\subset\mathcal{IR}^n}{\operatorname{argmax}} \; ||\nabla f(\mathbf{P})|| \\ \text{with} \;\; & f(\mathbf{S_i}) = L_s. \end{aligned} \tag{9}$$

Since f and $\nabla(f)$ are of dimension n, the constraint over the function value L_s effectively decreases the dimension of optimization to $(n-1)$. In other words, gradient surfing on the level set of a n-dimensional function amounts to optimizing (in this case, maximizing) a *different* $(n-1)$-dimensional function.

Applying this observation recursively, we can continue to decrease the dimensionality of the problem until the base level of dimension 2, for which the algorithm from Sect. 3.1 is applicable. Algorithm 3 describes this operation in pseudo code. A detailed algorithmic description is attached in "Appendix B").

Algorithm 3 Recursive Gradient Surfing (RGS)

Input: $f : \mathcal{IR}^n \rightarrow \mathcal{IR}$, $\mathbf{P_0}$
function **RGS**

1. If $n = 2$ apply Algorithm 1, return.
2. Otherwise
   (a) Decrease the dimensionality by setting $f_{n-1} \leftarrow -\|\nabla f_n\|^2$
   (b) **RGS** (f_{n-1},($p_1, p_2, \ldots, p_{n-1}$))
   (c) Estimate the parameter p_n using the function value constraint

end **RGS**

4 Numerical Evaluation

We demonstrate the performance of the proposed GS optimization method using a set of standard test functions and through an image processing application. The following subsections present these results.

4.1 Standard Benchmark Analysis

We first tested the GS method over a set of non-convex standard benchmark functions listed in Table 1 (cf. Cvijovic et al.. [21], Barhen et al.. [10] and Appendix, Fig. 9). The measure for comparison between the different methods is the number of objective function evaluations until the global optimum is obtained (achievement of 1% accuracy from the global minimum is considered as a suitable termination condition) or if the termination condition (in terms of a number of evaluations) has been met. For each test function, we report the averages measured from 100 independent runs, each initialized with a random initial guess. For all functions except H3, we used the 2D GS method described in Sect. 3.1. GS results for the H3 function, the only three-dimensional function in the standard benchmark, are shown for both the recursive approach and the iterative parameters approach. In *all* cases, including the evaluations in the following sections, we fixed the two parameters at $\alpha = 0.1,\ \beta = 0.1$. The code for the GS method, as well as our comparison code, is available publicly for the benefit of the community (http://icvl.cs.bgu.ac.il/gradient-surfing/). Although the GS method is based on the "steeper goes deeper" heuristic and does not guarantees convergence to the global optimum, 84% of the runs were successful in completing this task, showcasing the method reliability (denoted as success rate (SR) in Table 1). For each test function, the GS measure presented in Table 1 includes all 100 experiments, including the ones that did not convergence to the global optimum. Results in table indicate that the GS approach is clearly competitive and often outperforms the existing methods.

4.2 Application Benchmark—Optimization for Image Registration

It is always important to evaluate the effectiveness of a method in real-life practical application. To do so, we selected the problem of image registration, i.e., the process of matching a source image to a target image. Such registration is needed when the same scene is imaged from different cameras (or sensors), different viewpoints, times, distances, etc., [40–42], and it is a fundamental computation in computer vision, medical imaging, remote sensing, astrophotography, and numerous other fields. Although the registration objective functions are typically highly non-convex, to date, standard *local* optimization techniques are frequently used [40]. It is therefore constructive to examine how a new global optimization approach can perform on this challenge.

The registration problem is formulated as follows: Consider a *source* and *target* monochromatic images, each defined as $I : \Omega \subseteq \mathcal{IR}^2 \rightarrow \mathcal{IR}$ that maps every coordinate in a given (typically compact) domain Ω to pixel values in the range $[0, 1]$. Let $\mathcal{A}_{\mathbf{P}} : \Omega \rightarrow \Omega$ denote an affine registration transformation that maps every

Table 1 Evaluation of GS against existing global optimization methods in a standard optimization benchmark

Function	GP	RA	BR	SH	CA	H3	Success rate (%)
Dimension	2	2	2	2	2	3	
#Global minima	1	1	3	18	2	1	
#Local minima	4	50	4	760	6	4	
PRS [35]	5125	5964	4850	6700	–	5280	–
MS [36]	4400	–	1600	–	–	2500	–
SDE [37]	5439	–	2700	241,215	10,822	3416	–
EA [38]	460	2048	430	–	–	–	–
MLSL [39]	148	–	206	–	–	197	–
SA [26]	563	–	505	780	–	1459	–
GRDT [19]	5175	822	466	502	290	2811	–
TUN [16]	–	–	–	12,160	1469	–	–
TRUST [10]	103	**59**	55	**72**	31	**58**	–
HG-PSO [27]	> 2100	> 2100	> 2100	> 2100	–	> 2100	20
GR-PSO [27]	134	309	32	308	–	88	100
TS [21]	486	540	492	727	–	508	90
GS	**62**	201	**25**	184	**14**	72/ 86*	84

Following previous reports, performance is measured by the average number of function evaluations before the global minimum is obtained (minimal evaluations needed in bold). Data of prior methods were compiled from Cvijovic and Klinowski [21], Barhen et al. [10], and Chowdhury et al. [17], and Yong et al. [27]. SR denotes success rate. Unreported data are marked with "−," and ">" denotes cases where processing was terminated once number of evaluations exceeded a predefined threshold [27]. Functions tested include Goldstein-Price (GP), Rastrigin (RA), Branin (BR), Shubert (SH), Six-hump Camelback (CA), and Hartman (H3). Some of their properties are listed in the table

coordinate in the source image to a coordinate in the target image, and we seek to find the transformation that best matches the values of all mapped points. While the presented method is not restricted to any specific choice, for our empirical evaluation we used the popular L_2 norm as the cost function f. More formally, the cost function is defined by:

$$f = \sum_x \sum_y \|Target(x, y) - Source(\mathcal{A}_{\mathbf{P}}(x, y))\|^2 , \tag{10}$$

where $\mathbf{P}$ represents all possible parameters of the affine transformation $\mathcal{A}$ and $\mathcal{A}_{\mathbf{P}}$ is defined in homogeneous coordinates as

$$\mathcal{A}_{\mathbf{P}}(x, y) = \begin{bmatrix} a & b & t_h \\ c & d & t_v \\ 0 & 0 & 1 \end{bmatrix} , \tag{11}$$

where the four parameters (a, b, c, d) correspond to scaling, shearing, and rotation and (t_h, t_v) correspond to the horizontal and vertical translation. Ignoring sampling, quantization, and noise issues at this point, the globally minimum value of the cost function is zero and is obtained when the deformed source image has undergone a perfect transformation and the two images are fully aligned. We evaluated the performance of our GS optimization for the purpose of the registration task above using a random set of 100 natural images selected from five different classes (people, room, natural, space, and car) from the ImageNet dataset [43] that has been widely used for the evaluation of various computer vision tasks. The evaluation process included generating pairs of source and target images, where the latter is a random affine-transformed version of the former based on randomly selecting the 6 parameters of the transformation. The goal was to recover this transformation by globally minimizing the six-dimensional Eq. 10. To do so, we applied the iterative parameters GS method (Sect. 3.3) after initializing the optimization from a random starting points. This process was repeated 3 times for each image pair (i.e., for each unknown transformation), and the obtained registration was evaluated by applying it to the source image and comparing against the ground truth target image (Eq. 10).

We compared the performance of the GS method with 3 classical global optimization methods:[2] Simulated annealing (SA) [28], particle swarm optimization (PSO) [29,30], and genetic algorithm (GA) [31–33]. To ensure the fairness of the comparison, all methods were initialized with the same (random) starting points and were implemented using the default parameters supplied by MATLAB 2016 Global Optimization Toolbox while setting the two parameters required by GS to the default values $\alpha = \beta = 0.1$ (i.e., no effort to optimize these values was attempted).

To perform an equal-ground comparison when it comes to computational resources, all methods were forced to terminate and report their best result after a predefined number of cost function evaluations. More specifically, we checked how well did the

[2] Unlike other methods from Table 1, these methods were now selected for comparison due to their frequent use in various applications, ubiquitous and standard implementations, code availability, and their superiority to other both local and global optimization methods in locating the global minimum [44].

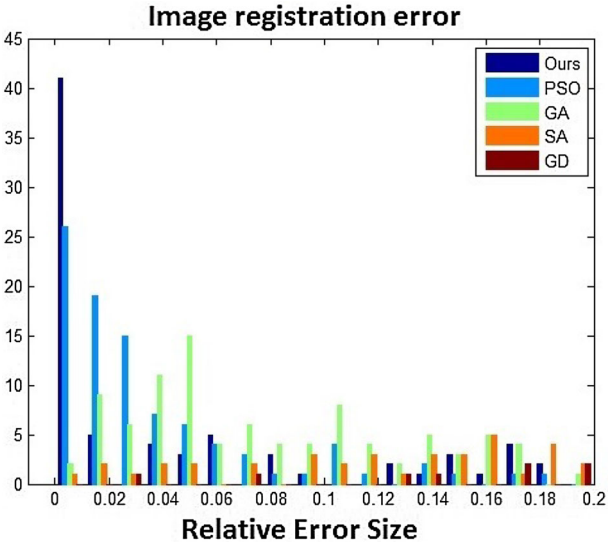


Fig. 4 Histogram of relative cost function error of our gradient surfing (GS) method in comparison with 3 different classical *global* optimization methods and to the *local* optimization GD method (whose results are mostly outside the selected range of relative errors)

classical optimization methods do when forced to terminate after the same number of evaluations it took our method to converge. The GS method detected the true global minimum in 41% of the experiments, while the classical GA, SA, and PSO methods detected it in only 2%, 1%, and 25% of the experiments, respectively (see Fig. 4).

When all methods were allowed to iterate until convergence, the GA and SA methods improved their detection rate to 25% and 13%, for a computational overhead that required 8 and 2 times more function evaluations than our GS solver, respectively. Only the PSO method performed better than the GS method, achieving 65% detection rate for a computational overhead of 2.

Interestingly, using the same step size ($\alpha = 0.1$) and the same initial guesses, GD obtained the global minimum in *none* of the trials, showing unequivocally the benefit of the leaping phase and the utility (as well as plausibility) of the "steeper goes deeper" heuristic for real-life non-convex problems.

4.3 Gaussian Mixture Model Benchmark

To further validate the proposed optimization method, we used a different set of non-convex functions in order to evaluate its performance. One important class of functions with much relevance for scientific work is the GMMs mentioned in Sect. 2, where the objective function is a sum of many multi-dimensional Gaussians, normalized to the range [0,100]. Evidence for successfully locating global minima on GMMs is of particular interest, given the great importance of GMMs for both basic and applicative research [45–47]. To use GMMs for the evaluation of global optimization techniques, we generated a large set of functions by pre-computing 100 3D GMM functions (as

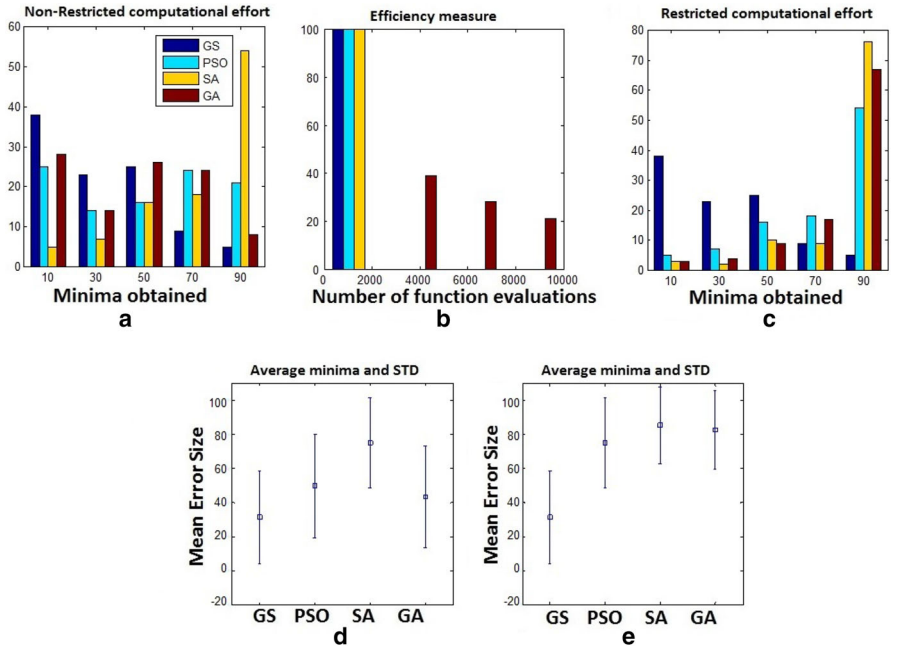


Fig. 5 Accuracy and efficiency for the three-dimensional GMM benchmark by our GS method versus three other commonly used classical global optimization methods. **a** Histogram of minima obtained when all methods continue to iterate until convergence. **b** Histogram of number of function evaluations for each method. **c** Histogram of minima obtained when all methods are restricted to the same computational effort. **d** Average minima and STD when all methods continue to iterate until convergence. **e** Average minima and STD when all methods are restricted to the same computational effort

mentioned in Sect. 2 and detailed in "Appendix A") and compared the minimal value obtained by applying our GS method. All other implementation and evaluation details were identical to Sect. 4.2.

Results indicate that the average of the minima obtained was 32 (STD = 27) for our proposed GS method, and 50 (STD = 30), 43 (STD = 29), and 75 (STD = 26) for the PSO, GA, and SA methods, respectively. Moreover, simply counting the number of cases where each method yields better result (i.e., deeper minima) than the competitors, we found that our proposed method outperformed PSO, GA, and SA in 68%, 63%, and 90% of trials, respectively.

Importantly, while the GS method outperforms other global optimization methods in terms of accuracy (minima obtained), it also requires significantly fewer function evaluations. The average number of function evaluations was 1100, 6700, and 1520 for the PSO, GA , and SA methods, respectively, but only 760 for ours.

While performing an equal-ground test as before, the PSO, GA, and SA methods produced an average minima of 75 (STD = 26), 83 (STD = 23), and 85 (STD = 22), respectively. While counting the number of cases where each method yields more accurate result, we found that our proposed method outperformed PSO, GA, and SA in 84%, 94%, and 90% of trials, respectively. Figure 5 presents the distribution

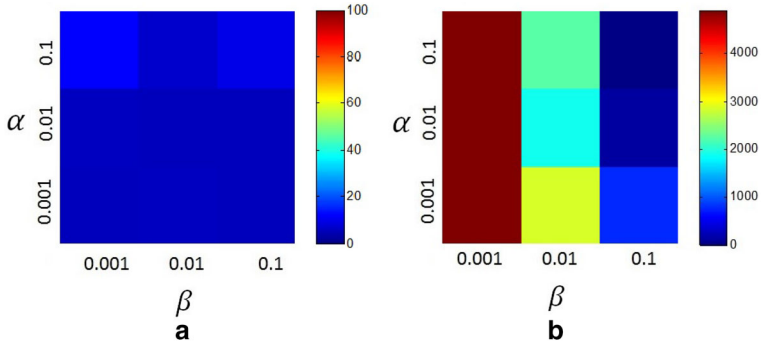


Fig. 6 Accuracy and efficiency for the GS method with different parameters. **a** Minima obtained with different α and β parameters. **b** Number of function evaluations with different α and β parameters

of accuracy and number of function evaluations from the three-dimensional GMM benchmark experiments.

4.4 Stability Analysis

In order to evaluate GS robustness to changes in parameter values, we ran 100 experiments using α and β with 3 different orders of magnitude ($\{0.001, 0.01, 0.1\}$) over a subset of GMM functions (cf. Sect. 2). To ensure the fairness of the comparison, each test was initialized with the same (random) starting points for every pair of parameters. While the results indicate the method to be stable for all α and β tested values in terms of accuracy (Fig. 6a), the computational resources required for detecting the global minima naturally increase when the step size values decreased. Since the level set search requires multiple function evaluation after every local minimization step, lower β values have greater influence on the required computational cost compared to lower α values (Fig. 6b). β values larger than 0.1 caused observable deviations from the level set during surfing.

5 Conclusions

In this work, we introduced the "steeper goes deeper" heuristic, implying that on each level set, greater gradients commonly point on a deeper minima; we verify this via numerous empirical experiments. Based on the proposed heuristic, we present a novel yet simple and efficient deterministic approach for multivariate global optimization. The proposed optimization method consists of two steps: a minimization step that locally minimizes the function value, and a leaping step that changes the current search point to a different point in the search space, chosen heuristically. We claim that moving in the new point direction decreases the probability of getting trapped in a local minimum.

We tested the proposed GS method performance and efficiency on a standard benchmark to find that, on the majority of cases, the GS usually outperforms other classical

methods in locating the global minimum while requiring less computational effort (measured by number of objective function evaluations). The generality and simplicity of the GS method and its advantage over the GD local optimization make it suitable to be adapted and integrated in numerous computational problems. While applying our method to an increased dimension registration problem, we found the results to be very satisfactory, applying it to larger-scale problems is left to future work.

Appendices

Appendix A: Soundness of the Heuristic

As described in Sect. 2, we empirically tested the soundness of our "steeper goes deeper" heuristic for several different families of non-convex functions, including 2D, 3D, and 4D GMMs, trigonometric and a mixture model of trigonometric and polynomial function families. This appendix provides additional details about these functions and results not reported in the main text.

The GMMs used for our tests included mixtures of 10 Gaussians in dimension $n \in \{2, 3, 4\}$.

$$G(x) = \sum_{i=1}^{10} \omega_i \cdot g_i(x). \tag{12}$$

In the 1D case, each component is defined as

$$g_i(x) = e^{-\frac{(x-\mu_i)^2}{2\cdot\sigma_i^2}}, \tag{13}$$

where μ_i is selected randomly in the range of $[-1, 1]$, ω_i is selected randomly in the range of $[0, 1]$, and σ_i is a uniform random variable in the interval $\sigma_i \in [0.2, 0.3]$. In the multi-dimensional case, each Gaussian is defined by

$$g_i(x) = e^{-\frac{1}{2}\cdot(\mathbf{x_i}-\boldsymbol{\mu_i})^T \Sigma_i^{-1}(\mathbf{x_i}-\boldsymbol{\mu_i})}, \tag{14}$$

where Σ is the diagonal covariance matrix, $\boldsymbol{\mu_i}$ is the nth-dimensional mean, and their scalar components are selected randomly as in the univariate case. One such non-convex function is shown in Fig. 7a.

A next class tested included trigonometric functions of the following form:

$$\sum_{n=1}^{10} sin(2 \cdot \pi \cdot \phi_n \cdot x) + cos(2 \cdot \pi \cdot \phi_n \cdot y), \tag{15}$$

where ϕ_n is a frequency selected randomly in the range of $[0, 0.1]$ and the domain is defined as $0 < x, y < 50$. One such non-convex trigonometric function is shown in Fig. 7b.

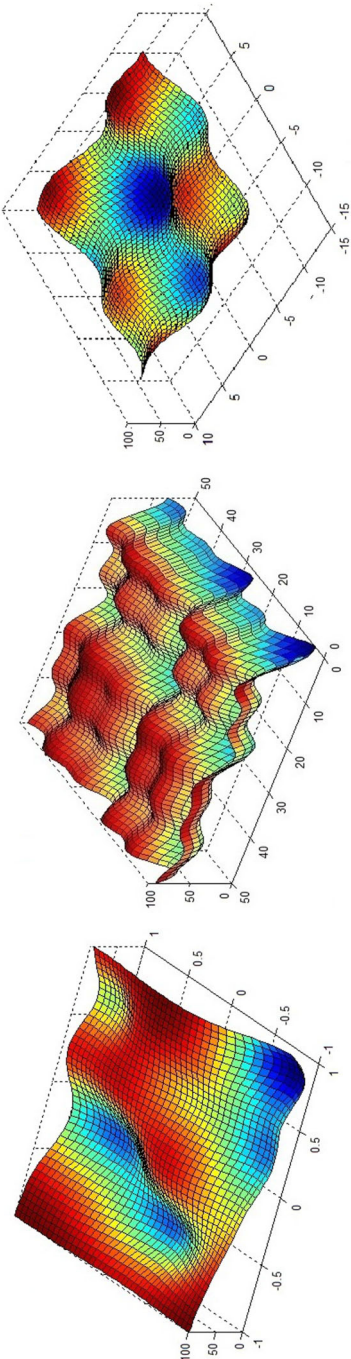


Fig. 7 Typical examples from our classes of evaluation functions. **a** A 2D GMM from our set. **b** A Trigonometric non-convex function from our set. **c** A mixture model of the trigonometric and polynomials non-convex function from our set

The last type of functions tested was a mixture model of trigonometric and fourth-order polynomial functions defined as follows:

$$\sum_{n=1}^{10} sin(2 \cdot \pi \cdot \phi_n \cdot x) + cos(2 \cdot \pi \cdot \phi_n \cdot y) + a_4 \cdot x^4 + a_3 \cdot x^3 - a_2 \cdot x^2 + a_1 \cdot x, \quad (16)$$

where a_1 and a_2 were selected randomly in the range of $[0, 1]$, a_3 and a_4 selected randomly in the range of $[0, 0.01]$, and ϕ_n is a frequency selected randomly in the range of $[0, 0.1]$. The domain in this case was set to $-10 < x, y < 10$, and one such function is shown in Fig. 7c. In addition to the results shown in Fig. 3, we further evaluated the "steeper goes deeper" heuristics by the following steps. Given a test function, we first segmented selected level sets to two subsets—the one that belonged to the global BOA (cf. Sect. 2) and the second that complemented it. In each subset, we located the point of maximum gradient magnitude, with $\mathbf{p_g}$ denoting the point of maximum gradient in the global BOA and $\mathbf{p_l}$ the corresponding point in the local BOAs. Finally, we compared the gradient magnitude between these two points. If our heuristics is sound, the gradient magnitude at $\mathbf{p_g}$ should be greater than $\mathbf{p_l}$ more often than not.

More formally, we examined the distribution of the values in the function

$$\Phi(L_s) = \left\| \nabla f(\mathbf{p_g}) \right\| - \left\| \nabla f(\mathbf{p_l}) \right\|, \quad (17)$$

where L_s denotes a level set for which we identify the two points

$$\mathbf{p_g} = \underset{p}{\text{argmax}} ||\nabla f(\mathbf{p})|| \text{ subject to } \{ f(\mathbf{p}) = L_s \ , \ \mathbf{p} \in \text{ global BOA } \} \quad (18)$$

$$\mathbf{p_l} = \underset{p}{\text{argmax}} ||\nabla f(\mathbf{p})|| \text{ subject to } \{ f(\mathbf{p}) = L_s \ , \ \mathbf{p} \notin \text{ global BOA } \}. \quad (19)$$

If our heuristics is sound, $\Phi(L_s)$ should be biased toward positive values (Fig. 8).

Figure 9 presents the histogram of the Φ values taken from 5 different level sets $\{20, 30, 40, 60, 80\}$ using 100 different functions in each of the different classes. In all cases, we clearly find that the "steeper goes deeper" heuristic is significantly more often valid than not (note the strong bias toward positive values).

Appendix B: Algorithmic Description of Recursive Gradient Surfing (RGS)

The recursive gradient surfing (RGS) operates on a given function f_d from a given initial guess $\mathbf{P_0}$. At the heart of the approach is a progressive reduction in the dimension of the optimization (until a base dimension is obtained) with a corresponding change in the objective function. For this reason, the dimension of the optimization d also is provided as input to the function. A detailed algorithmic description is presented next.

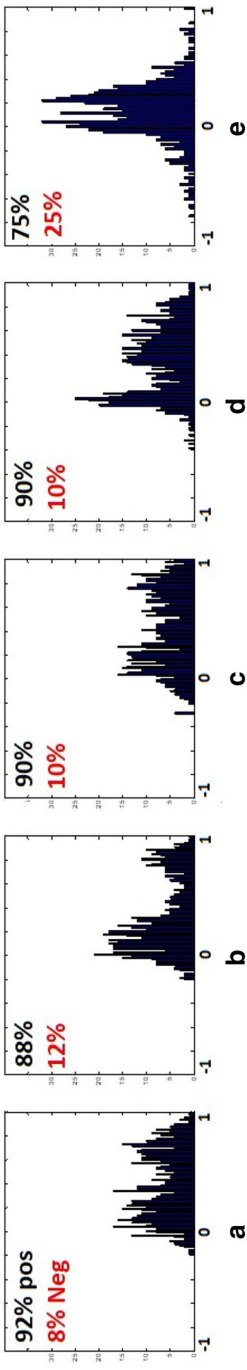


Fig. 8 Histogram of Φ values for our different non-convex test function classes. Insets show the percentage of positive (top) and negative (bottom, red) values; positive values support the soundness of the "steeper goes deeper" heuristic. **a** Φ values for 2D GMM functions. **b** Φ values for 3D GMM functions. **c** Φ values for 4D GMM functions. **d** Φ values for trigonometric functions. **e** Φ values for a mixture of the trigonometric functions and fourth-order polynomials

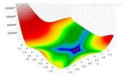
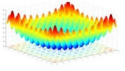
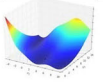
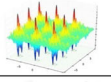
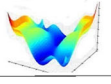
Name (Abbreviation)	Reference	Math expression	Domain	Range	Plot	Number of Global minima	Number of Local Minima
Goldstein-Price (GP)	Cvijovic et al 1995	$f(x,y) = [1+(x+y+1)^2 \cdot (19-14x+3x^2-14y+6xy+2y^2)]$ $\cdot[30+(2x-3y)^2(18-32x+12x^2+48y-36xy+27y^2)])$	$-2 \le x,y \le 2$	$-3.12 \le f \le 2.11$		1	4
Rastrigin (RA)	Cvijovic et al 1995	$f(x,y) = x^2 + y^2 - cos(18x) - cos(18y)$	$-2 \le x,y \le 2$	$-2 \le f \le 3.71$		1	50
Branin (BR)	Cvijovic et al 1995	$f(x,y) = [y-(5.1/4\pi^2)x^2+(5/\pi)x-6]^2+$ $10(1-1/8\pi)cos(x)+10$	$-5 \le x \le 15$ $0 \le y \le 15$	$0.3979 \le f \le 19.6$		3	4
Shubert (SH)	Cvijovic et al 1995	$f(x,y) = \left\{\sum_{j=1,..,5}(j \cdot cos[(j+1)x+j])\right\} \cdot \left\{\sum_{j=1,..,5}(j \cdot cos[(j+1)y+j])\right\}$	$-10 \le x,y \le 10$	$-186.7035 \le f \le -0.02$		18	760
Six-hump Camelback (CA)	Bar- Hen et al 1995	$f(x,y) = (4-2.1 \cdot x^2 + x^4/3) \cdot x^2 + x \cdot y + (-4+4 \cdot y^2)y^2$	$-3 \le x \le 3$ $-2 \le y \le 2$	$-1.03 \le f \le 1.2$		2	6
Hart 3 (H3)	Cvijovic et al 1995	$f(x_1,x_2,x_3) = \sum_{i=1,..,4} \alpha_i \cdot exp(-\sum_{j=1,..,3} A_{i,j} \cdot (x_i - P_{i,j})^2)$	$0 \le x_1,x_2,x_3 \le 1$	$-3.8627 \le f \le 0$		1	4

Fig. 9 Test functions used for the standard benchmark for global optimization

Algorithm 4 Recursive Gradient Surfing (RGS)

$(p_1, \ldots, p_d)$ =function **RGS**$(d, \mathbf{P_0}, f_d)$ returns estimation of global optimum
if $d > 2$ **then**
  $i \leftarrow 0$
  repeat
    $i \leftarrow i + 1$ % Index for current guess in each % level of the recursion.
    $\mathbf{P_i} \leftarrow \mathbf{P}_{i-1} + \alpha \cdot \nabla f_d(\mathbf{P}_{i-1})$ % GD step $\mathbf{P_i} = (p_1, p_2, \ldots, p_{d-1}, p_d)_i$.
    $V_{ls} \leftarrow f_d(\mathbf{P_i})$ % Current level set value.
    $f_{d-1} \leftarrow -\|\nabla f_d\|^2$ % Set the new cost function.
    $\mathbf{P_{0,d}} \leftarrow (p_1, p_2, \ldots, p_{d-1})_i$ % Set new initial guess.
    $\mathbf{P_i} \leftarrow$**RGS** $(d-1, \mathbf{P_{0,d}}, f_{d-1})$ % Decrease the dimensionality of % the problem until the base % level of dimension 2.
    Find the point of maximal gradient p_d^{surf} on the level set V_{ls}
    $\mathbf{P_i} \leftarrow \mathbf{P_{0,d}} \oplus p_d^{surf}$ % Change the current guess % according to the heuristic.
  until {Termination conditions for GD method is satisfied}
else{ **d=2** base case for 2 dimensions}
  Estimate (p_1, p_2) =**GS**$(f_d, \mathbf{P_0})$ by the $2D$ Surfing Method.
end if

Apart from the recursive call, a key step in the algorithm is the need to find the point of maximal gradient on the level set V_{ls}. The level set itself is defined by the rootlike constraint $f_d(\mathbf{P_{0,d}} \oplus p_d^{surf}) - V_{ls} = 0$, a one-dimensional problem that may be solved with any root finding techniques such as Newton Raphson. The corresponding 1D (non-convex) optimization problem (i.e., the root with maximal gradient magnitude) may be addressed with many of the tunneling approaches [16]

References

1. Ghahramani, Z.: Probabilistic machine learning and artificial intelligence. Nature **521**(7553), 452–459 (2015)
2. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, Berlin (2010)
3. Jenkinson, M., Smith, S.: A global optimisation method for robust affine registration of brain images. Med. Image Anal **5**(2), 143–156 (2001)
4. Ben-Yosef, G., Ben-Shahar, O.: Curve completion as minimum action in the primary visual cortex. J. Vis. **10**(7), 1165–1165 (2010)
5. Goffe, W.L., Ferrier, G.D., Rogers, J.: Global optimization of statistical functions with simulated annealing. J. Econom. **60**(1–2), 65–99 (1994)
6. Wales, D.J., Scheraga, H.A.: Global optimization of clusters, crystals, and biomolecules. Science **285**(5432), 1368–1372 (1999)
7. Lam, A.Y., Li, V.O.: Chemical reaction optimization: a tutorial. Memet. Comput. **4**(1), 3–17 (2012)
8. Xue, D., Balachandran, P.V., Hogden, J., Theiler, J., Xue, D., Lookman, T.: Accelerated search for materials with targeted properties by adaptive design. Nat. Commun. **7** (2016)
9. Hartmann, A.K., Rieger, H.: Optimization Algorithms in Physics, vol. 2. Citeseer (2002)
10. Barhen, J., Protopopescu, V., Reister, D.: Trust: a deterministic algorithm for global optimization. Science **276**(5315), 1094–1097 (1997)
11. Li, Z., Scheraga, H.A.: Monte carlo-minimization approach to the multiple-minima problem in protein folding. Proc. Natl. Acad. Sci. **84**(19), 6611–6615 (1987)
12. Michalewicz, Z., Dasgupta, D., Le Riche, R.G., Schoenauer, M.: Evolutionary algorithms for constrained engineering problems. Comput. Ind. Eng. **30**(4), 851–870 (1996)

13. Horst, R., Pardalos, P.M.: Handbook of Global Optimization, vol. 2. Springer, Berlin (2013)
14. Dzemyda, G., Saltenis, V., Zilinskas, A.: Stochastic and Global Optimization. Series on Nonconvex Optimization and its Applications (2002)
15. Horst, R., Pardalos, P.M., Van Thoai, N.: Introduction to Global Optimization. Springer, Berlin (2000)
16. Levy, A., Montalvo, A.: The tunneling algorithm for the global minimization of functions. SIAM J. Sci. Stat. Comput. **6**(1), 15–29 (1985)
17. Chowdhury, P.R., Singh, Y.P., Chansarkar, R.: Hybridization of gradient descent algorithms with dynamic tunneling methods for global optimization. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **30**(3), 384–390 (2000)
18. Yao, Y.: Dynamic tunneling algorithm for global optimization. IEEE Trans. Syst. Man Cybern. **19**(5), 1222–1230 (1989)
19. Singh, Y.P., RoyChowdhury, P.: Dynamic tunneling based regularization in feedforward neural networks. Artif. Intell. **131**(1), 55–71 (2001)
20. Lampariello, F., Liuzzi, G.: A filling function method for unconstrained global optimization. Comput. Optim. Appl. **61**(3), 713–729 (2015)
21. Cvijovic, D., Klinowski, J.: Taboo search: an approach to the multiple minima problem. Science **267**(5198), 664 (1995)
22. Hwang, C.R.: Simulated annealing: theory and applications. Acta Appl. Math. **12**(1), 108–111 (1988)
23. Suman, B., Kumar, P.: A survey of simulated annealing as a tool for single and multiobjective optimization. J. Oper. Res. Soc. **57**(10), 1143–1160 (2006)
24. Fouskakis, D., Draper, D.: Stochastic optimization: a review. Int. Stat. Rev. **70**(3), 315–349 (2002)
25. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: a survey. Oper. Res. **14**(4), 699–719 (1966)
26. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simmulated annealing. Science **220**(4598), 671–680 (1983)
27. Wang, Y.J.: Improving particle swarm optimization performance with local search for high-dimensional function optimization. Optim. Methods Softw. **25**(5), 781–795 (2010)
28. Ingber, L.: Adaptive simulated annealing (asa): lessons learned. Control Cybern. **25**, 33–54 (1996)
29. Eberhart, R.C., Kennedy, J., et al.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, vol. 1, pp. 39–43. New York, NY (1995)
30. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of Machine Learning, pp. 760–766. Springer, Berlin (2011)
31. Conn, A., Gould, N., Toint, P.: A globally convergent lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. Math. Comput. Am. Math. Soc. **66**(217), 261–288 (1997)
32. Conn, A.R., Gould, N.I., Toint, P.: A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. SIAM J. Numer. Anal. **28**(2), 545–572 (1991)
33. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Aaddison-Wesley Professional, Reading (1989)
34. Sagan, H.: Hilberts space-filling curve. In: Space-Filling Curves, pp. 9–30. Springer, Berlin (1994)
35. Anderssen, R.S., Jennings, L.S., Ryan, D.M.: Optimization (1972)
36. Dekkers, A., Aarts, E.: Global optimization and simulated annealing. Math. Program. **50**(1–3), 367–393 (1991)
37. Aluffi-Pentini, F., Parisi, V., Zirilli, F.: Global optimization and stochastic differential equations. J. Optim. Theory Appl. **47**(1), 1–16 (1985)
38. Yong, L., Lishan, K., Evans, D.J.: The annealing evolution algorithm as function optimizer. Parallel Comput. **21**(3), 389–400 (1995)
39. Boggs, P.T., Byrd, R.H., Schnabel, R.B., et al.: Numerical Optimization 1984: Proceedings of the SIAM Conference on Numerical Optimization, Boulder, Colorado, June 12–14, 1984, vol. 20. SIAM (1985)
40. Zitova, B., Flusser, J.: Image registration methods: a survey. Image Vis. Comput. **21**(11), 977–1000 (2003)
41. Goshtasby, A.A.: 2-D and 3-D Image Registration: For Medical, Remote Sensing, and Industrial Applications. Wiley, London (2005)
42. Wyawahare, M.V., Patil, P.M., Abhyankar, H.K.: Image registration techniques: an overview. Int. J. Signal Process. Image Process. Pattern Recognit. **2**(3), 11–28 (2009)

43. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 248–255. IEEE (2009)
44. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**(3), 1247–1293 (2013)
45. Zivkovic, Z.: Improved adaptive gaussian mixture model for background subtraction. In: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, vol. 2, pp. 28–31. IEEE (2004)
46. Bilmes, J.A.: A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Int. Comput. Sci. Inst. **4**(510), 126 (1998)
47. Bishop, C.M.: Pattern recognition. Machine Learning **128** (2006)