algorithms

Article

A Direct Search Algorithm for Global Optimization

Enrique Baeyens [1,*], Alberto Herreros [1] and José R. Perán [2]

[1] Instituto de las Tecnologías Avanzadas de la Producción (ITAP), Universidad de Valladolid, Paseo del Cauce 59, 47011 Valladolid, Spain; albher@eii.uva.com
[2] Centro Tecnológico Cartif, Parque Tecnológico de Boecillo 205, 47151 Bocillo, Spain; peran@cartif.es
* Correspondence: enrbae@eii.uva.es; Tel.: +34-983-423-909

Academic Editor: George Karakostas
Received: 19 January 2016; Accepted: 7 June 2016; Published: 13 June 2016

Abstract: A direct search algorithm is proposed for minimizing an arbitrary real valued function. The algorithm uses a new function transformation and three simplex-based operations. The function transformation provides global exploration features, while the simplex-based operations guarantees the termination of the algorithm and provides global convergence to a stationary point if the cost function is differentiable and its gradient is Lipschitz continuous. The algorithm's performance has been extensively tested using benchmark functions and compared to some well-known global optimization algorithms. The results of the computational study show that the algorithm combines both simplicity and efficiency and is competitive with the heuristics-based strategies presently used for global optimization.

Keywords: global optimization; direct search methods; search space transformation; derivative-free optimization; heuristics-based optimization

1. Introduction

An optimization problem consists of finding an element of a given set that minimizes (or maximizes) a certain value associated with each element of the set. In this paper, we are interested in minimizing the value of a real valued function defined over the n-dimensional Euclidean space. Without loss of generality, our problem is equivalent to minimizing a real valued function over the open unit n-box $(0,1)^n \subset \mathbb{R}^n$.

Let f denote a real function defined over the open unit n-box $(0,1)^n \subset \mathbb{R}^n$, and consider the following optimization problem:

$$\min\{f(\mathbf{x}) : \mathbf{x} \in (0,1)^n\} \tag{1}$$

The function to be minimized f is called the cost function and the unit n-box $(0,1)^n$ is the search space of the problem. Our aim is to obtain a point $\mathbf{x}^* \in (0,1)^n$, such that the cost function f attains a minimum at $\mathbf{x}^*$, *i.e.*, $f(\mathbf{x}^*) \leq f(x)$, $\forall \mathbf{x} \in (0,1)^n$. We shall make the following assumption:

A1 The cost function f attains its minimum at a point of the search space.

If the function f is continuously differentiable in $(0,1)^n$, then:

$$\nabla f(\mathbf{x}^*) = 0,\ \mathbf{x}^* \in (0,1) \tag{2}$$

Many optimization methods make use of the stationarity Equation (2) to compute candidate optimizer points. These methods require the explicit knowledge of the gradient, but they do not guarantee that the point obtained is an optimizer, except for a pseudoconvex function.

We shall also make an additional assumption that will restrict our alternatives to design a procedure to compute the optimum.

A2 The gradient of the cost function is not available for the optimization mechanism.

Consequently, only function evaluations can be used to compute the minimum.

Remark 1. The problem Equation (1) is equivalent to unconstrained optimization. Consider the following unconstrained optimization problem:

$$\min\{f(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \mathbb{R}^n\} \tag{3}$$

and let $\boldsymbol{\xi} = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_n \end{bmatrix}^T$ be the vector of variables of the cost function. The invertible transformation:

$$x_i = \frac{1}{1 + e^{-\xi_i}} \tag{4}$$

transforms the real line into the unit interval. Consequently, we can convert an unconstrained minimization problem into an open unit n-box constrained minimization problem by transforming each of its variables. Similarly, the linear transformation:

$$x_i = \frac{\xi_i - a}{b - a} \tag{5}$$

converts the open interval (a, b) into the open unit interval. Consequently, both unconstrained and arbitrary open n-box optimization problems are included in our formulation. Furthermore, under Assumption A2 and by choosing appropriate values of a and b, the global minimum of function f is attained in an interior point of the open n-box $(a, b)^n$.

The rest of this paper is organized as follows. Section 2 is a review of the literature on direct search and global optimization algorithms. The direct search algorithm for global optimization is designed in Section 3. The algorithm makes use of a set of simplex-based operations for a transformed cost function. We begin by reviewing some basic results about n-dimensional simplices that will be used later to define the algorithm and study its properties. Then, we will introduce a transformation of the cost function that is crucial to improving the exploratory properties of the algorithm. Finally, we conclude this section by explaining the algorithm in detail and analyzing its convergence properties. In Section 4, an experimental study of the algorithm's performance is accomplished by using three well-known test functions. Its performance is also compared to other global optimization strategies. The conclusions are presented in Section 5.

2. Direct Search Methods and Global Optimization

A direct search method solves optimization problems without requiring any information about the gradient of the cost function [1,2]. Unlike many traditional optimization algorithms that use information about the gradient or higher derivatives to search for an optimal point, a direct search algorithm searches a set of points around the current point, looking for one where the value of the cost function is lower than the value at the current point. Direct search methods solve problems for which the cost function is not differentiable or not even continuous. Direct search optimization has been receiving increasing attention over recent years within the optimization community, including the establishment of solid mathematical foundations for many of the methods considered in practice [1,3–7]. These techniques do not attempt to compute approximations to the gradient, but rather, the improvement is derived from a model of the cost function or using geometric strategies. There exists a great number of problems in different fields, such as engineering, mathematics, physics, chemistry, economics, medicine, computer science or operational research, where derivatives are not available, but there is a need for optimization. Some relevant examples are: tuning of algorithmic parameters, automatic error analysis, structural design, circuit design, molecular geometry or dynamic pricing. See Chapter 1 of [6] for a detailed description of the applications.

Initially, direct search methods have been dismissed for several reasons [8]. Some of them are the following: they are based on heuristics, but not on rigorous mathematical theory; they are slow to converge and only appropriate for small-sized problems; and finally, there were no mathematical analysis tools to accompany them. However, these drawbacks can be refuted today. Regarding slow convergence, in practice, one only needs improvement rather than optimality. In such a case, direct methods can be a good option because they can obtain an improved solution even faster than local methods that require gradient information. In addition, direct search methods are straightforward to implement. If we not only consider the elapsed time for a computer to run, but the total time needed to formulate the problem, program the algorithm and obtain an answer, then a direct search is also a compelling alternative. Regarding the problem size, it is usually considered that direct search methods are best suited for problems with a small number of variables. However, they have been successfully applied to problems with a few hundred variables. Finally, regarding analytical tools, several convergence analysis studies for a direct search algorithm were published starting in the early 1970s [9–12]. These results prove that it is possible to provide rigorous guarantees of convergence for a large number of direct search methods [4], even in the presence of heuristics. Consequently, direct search methods are regarded today as a respectable choice, and sometimes the only option, for solving certain classes of difficult and practical optimization problems, and they should not be dismissed as a compelling alternative in these cases. Direct search methods are broadly classified into local and global optimization approaches.

2.1. Local Optimization Approaches

Since there does not exist a suitable algorithmic characterization of global optima, most optimizers are based on local optimization. The most important derivative-free local based methods are pattern search methods and model-based search methods. Pattern search methods perform local exploration of the cost function on a pattern (a set of points conveniently selected). The Hooke–Jeeves [13] method and the Nelder–Mead simplex method [14] are two well-known examples of pattern search local methods.

The method of Hooke and Jeeves consists of a sequence of exploratory moves from a base point in the coordinate directions, which, if successful, are followed by pattern moves. The purpose of an exploratory move is to acquire information about the cost function in the neighborhood of the current base point. A pattern move attempts to speed up the search by using the information already acquired about the function to identify the best search direction.

The Nelder–Mead simplex algorithm belongs to the class of simplex-based direct search methods introduced by Spendley *et al.* in [15]. This class of methods evolves a pattern set of $n+1$ vectors on an n-dimensional search space that is interpreted as the vertex set of a simplex, *i.e.*, a convex n-dimensional polytope. Different simplex-based methods use different operators to evolve the vertex set. The Nelder–Mead algorithm starts by ordering the vertices of the simplex and uses five operators: reflection, expansion, outer contraction, inner contraction and shrinkage. At each iteration, only one of these operations is performed. As a result, a new simplex is obtained such that either contains a better vertex (a vertex that is better than the worst one and substitutes it) or has a smaller volume. From the point of view of the function value, every operator, except for the shrinkage, is productive and obtains a vertex that improves the worst one. The shrinkage operator reduces the volume of the simplex to guarantee algorithm termination. Unfortunately, the convergence of the Nelder–Mead simplex algorithm to a local minimizer is not guaranteed [16], and a number of modifications have been introduced to get convergence. Nevertheless, the Nelder–Mead simplex algorithm remains as one of the most popular direct search algorithms.

Model-based search methods are based on the idea of using function evaluations to compute a model of the cost function and to obtain derivative approximations from this model. Many model-based methods are trust-region algorithms [17] that interpolate the cost function in some region of appropriate shape and size to obtain a good approximation of the cost function that can later be easily optimized on that region. Typically, the model to be adjusted is a fixed-order polynomial, and the trust region is a sphere in some norm of the search space. The model function is sequentially optimized in the trust region, and the trust region is updated for the next iteration until the approximated model satisfies a first-order optimality condition. Model-based methods also include [18–20]. During the last decade, there has been a considerable amount of work to handle constraints in the field of direct search methods. New algorithms have been developed to deal with bounds and linear inequalities [21,22] smooth nonlinear constraints [23,24] or non-smooth constraints [25–27]. In this work, we do not deal with constraints. Our aim here is to develop a simple and efficient algorithm that improves the heuristics-based optimization methods that are presently used to obtain the global minimum in a multidimensional and multimodal continuous function.

2.2. *Global Optimization Approaches*

Local optimization methods are the best option for convex, or at least unimodal, optimization problems. However, most real-world problems are not convex, and the solution provided by a local method is not guaranteed to be global. Unfortunately, even if derivatives are available, it is not possible in general to prove that a local optimum is global. A rigorous global optimization algorithm for a multimodal cost function requires in-depth exploration of the search space, and this is only possible for functions with a small number of variables, because the computational time for space exploration increases exponentially with the dimension of the search space.

In order to circumvent the curse of dimensionality, many global optimization algorithms use random search [28] and heuristics [29,30]. Some successful classes of global optimization algorithms are simulated annealing algorithms [31], evolutionary algorithms [32], estimation of distribution algorithms [33], particle swarm optimization algorithms [34,35], differential evolution algorithms [36,37], tabu search algorithms [38,39] or ant colony optimization algorithms [40,41].

In recent years, there has been a great interest in derivative-free methods for global optimization; see the monographs [42–46] and the references therein. In particular, there are two classes of algorithms that are of special interest. The first class is Lipschitz optimization methods. Lipschitz methods substitute the cost function by an auxiliary function that underestimates it. The auxiliary function is linear piecewise and is obtained by partitioning the search space and repeatedly using the Lipschitz property [47–50]. An interesting Lipschitz method is DIRECT (DIviding RECTangles), that was introduced in [51]. In this method, the cost function is evaluated at several sample points by using all possible weights on local *versus* global search expressed by the Lipschitz constant. The DIRECT algorithm has gained popularity due to its simplicity. However, it has two weaknesses. First, it is slow in reaching the solution with high accuracy, and second, it spends an excessive number of evaluations exploring local minima. Some variants have been developed to improve these weaknesses [52–55].

Another class of global optimization methods uses a filled function to avoid local minima. The basic idea is to design an auxiliary function, called the filled function, such that when the algorithm reaches a local minimum, the minimization of the filled function provides a point that moves away from the basin of the local minimum. The filled function method was introduced by Ge [56,57]. It has had increasing interest in filled methods in the last few years [58–60].

3. A Direct Search Algorithm for Global Optimization

The direct search algorithm for global optimization proposed here has two main ingredients: a transformed cost function and a simplex-based mechanism to evolve the initial point. The transformed cost function improves global exploration, while the evolution mechanism guarantees the termination and global convergence to a stationary point. To the best of our knowledge, the transformed cost function has not been previously used in the literature, and it is a key ingredient of our approach, because it provides global exploration of the search space. The new algorithm is an efficient alternative to heuristics-based optimization methods for global optimization and preserves the convergence properties of a well-designed direct search method.

The algorithm makes use of an initial point that is allocated as a qualified vertex of an n-dimensional initial simplex. This simplex is evolved by a sequence of operations that depend on the value of the transformed cost function at each vertex. Each operation produces a similar simplex of a different size. The sequence of operations is designed in such a way that the simplex tends to asymptotically collapse in a single point. The algorithm terminates when the simplex vertices are close enough to each other.

Before explaining the algorithm and its operators in detail, we shall review some basic geometric results about n-simplices. Later, we shall introduce the cost function transformation and its main properties.

3.1. Notation

A brief summary of the notation used in this paper is included next for quick reference.

Set of integer numbers in interval $\mathcal{I}$: $\mathbb{Z}_{\mathcal{I}} := \{z \in \mathbb{Z} : z \in \mathcal{I}\}$.
Closed convex hull of a set $\mathcal{A}$: $\mathbf{Co}(\mathcal{A})$.
Difference set $\mathcal{A} \backslash \mathcal{B} = \{x \in \mathcal{A} : x \notin \mathcal{B}\}$.
Larger integer less than or equal to x: $\lfloor x \rfloor$.
Inner product of Euclidean space: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$.
p-norm of Euclidean space: $\|\mathbf{x}\|_p = \left(\sum\limits_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, p \in \mathbb{Z}_{[1,\infty)}$.
∞-norm of Euclidean space: $\|\mathbf{x}\|_\infty = \lim\limits_{p \to \infty} \|\mathbf{x}\|_p = \max\{|x_i| : \forall i\}$.

3.2. Review of the Basic Geometric Results about n-Simplices

Let $\mathcal{V} = \{\mathbf{x}_i : i \in \mathbb{Z}_{[0,n]}\}$ be a set of $n+1$ affinely-independent points in $\mathbb{R}^n$, then the closed convex hull of $\mathcal{V}$ is an n-simplex Σ with vertex set $\mathcal{V}$, and it is denoted as follows:

$$\Sigma = \mathbf{Co}(\mathcal{V}) \tag{6}$$

Next, we introduce certain specific types of n-simplices that are non-standard in the literature.

Definition 1 (Right n-simplex)**.** A right n-simplex has a vertex, such that the n edges intersecting at it are pairwise orthogonal. This vertex is called the right vertex.

Definition 2 (Isosceles right n-simplex)**.** A right n-simplex, such that every edge intersecting at the right vertex has the same length Δ, is an isosceles right n-simplex of length Δ.

Definition 3 (Standard n-simplex)**.** The standard n-simplex is an isosceles right n-simplex of unit length.

By locating the zero of $\mathbb{R}^n$ in the right vertex, the standard n-simplex Σ_n^0 is the n-dimensional polytope with vertex set $\mathcal{V} = \{\mathbf{b}_i \in \mathbb{R}^n : i \in \mathbb{Z}_{[0,n]}\}$ where $\mathbf{b}_0$ is the zero vector, and $\mathbf{b}_i$ is the i-th element of the standard basis of $\mathbb{R}^n$. Alternatively, the standard n-simplex can be defined as the intersection of the closed unit ball in the one-norm of $\mathbb{R}^n$ and the non-negative orthant, *i.e.*, $\Sigma_n^0 = \{\mathbf{x} \in [0,1]^n : \|\mathbf{x}\|_1 \leq 1\}$.

The content of an n-dimensional object is the measure of the amount of space inside its boundary. For instance, the content of a two-simplex is its area, and the content of a three-simplex is its volume. It can be proven by induction that the content of a standard n-simplex is $\frac{1}{n!}$.

A regular n-box is an n-dimensional box with edges of equal length. If the content of an n-simplex is not zero, then a geometric object of nonzero content can fit in its interior. The following lemma provides the edge length of the maximum regular n-box that can fit in the interior of a standard n-simplex.

Lemma 1. *The regular n-box of maximum size that can be contained inside a standard n-simplex has edge length $\frac{1}{n}$ and content $\left(\frac{1}{n}\right)^n$.*

Proof. The standard n-simplex is given as $\Sigma_n^0 = \{\mathbf{x} \in [0,1]^n : \|\mathbf{x}\|_1 \leq 1\}$. The maximum n-box that can be allocated inside the standard n-simplex has a vertex in the origin, and it is oriented along the coordinate directions. The opposite vertex to the origin of this regular n-box has coordinates $x_i = \frac{1}{n}$ for $i \in \mathbb{Z}_{[1,n]}$; therefore, its edge length is $\frac{1}{n}$, and its content is $\left(\frac{1}{n}\right)^n$. □

The edges that intersect at the right vertex of a standard n-simplex are oriented along the coordinate directions, so these edge directions define a nonnegative orthant. Any direction inside this orthant can be expressed by a nonnegative n-dimensional vector of unit norm, *i.e.*, $\langle \mathbf{d}, \mathbf{d} \rangle = 1$. A direction $\mathbf{d}$ forms angles θ_i with the coordinate directions whose cosines are called the directional cosines and are given by $\cos\theta_i = \langle \mathbf{d}, \mathbf{b}_i \rangle$. The following lemma proves that any direction $\mathbf{d}$ contained in the nonnegative orthant has at least one directional cosine that never is less than $\frac{1}{\sqrt{n}}$.

Lemma 2. *The smallest directional cosine of any direction contained in the orthant formed by the edges intersecting at the right vertex of a standard n-simplex is never less than $\frac{1}{\sqrt{n}}$.*

Proof. Without loss of generality, suppose that the right vertex is the zero point and that the edges are in the coordinate directions, characterized by the elements of the standard basis of $\mathbb{R}^n$, *i.e.*, $\mathbf{b}_i$ for $i \in \mathbb{Z}_{[0,n]}$. Consider the direction $\mathbf{d}^* = \begin{bmatrix} \frac{1}{\sqrt{n}} & \cdots & \frac{1}{\sqrt{n}} \end{bmatrix}$ whose directional cosines are $\cos\theta_i^* = \langle \mathbf{d}^*, \mathbf{b}_i \rangle = \frac{1}{\sqrt{n}}$ for all $i \in \mathbb{Z}_{[1,n]}$. Suppose that unlike the lemma statement, there exists a direction $\mathbf{d}$ with unit norm $\langle \mathbf{d}, \mathbf{d} \rangle = 1$, such that $\cos\theta_i = \langle \mathbf{d}, \mathbf{b}_i \rangle < \frac{1}{\sqrt{n}}$ for all $i \in \mathbb{Z}_{[1,n]}$, then $\langle \mathbf{d}, \mathbf{d} \rangle < n \sum_{i \in \mathbb{Z}_{[1,n]}} \frac{1}{n} = 1$. This contradicts the fact that $\mathbf{d}$ has unit norm, and the lemma is proven. □

A consequence of this lemma is that any direction contained in the nonnegative n-orthant always forms an angle less than $\frac{\pi}{2}$ radians with at least one of the coordinate directions that form the orthant. In addition, if we consider an arbitrary direction $\mathbf{d} \in \mathbb{R}^n$, then there always exists an n-simplex similar to the standard n-simplex, such that the edge directions of its right vertex define an orthant, such that the smallest directional cosine is never less than $\frac{1}{\sqrt{n}}$. This n-simplex is obtained by rotating π radians the edges of the standard n simplex corresponding to the negative components of the direction vector $\mathbf{d}$.

3.3. Function Transformation

A key ingredient of our direct search global optimization algorithm is a transformation of the cost function that improves its exploratory features, providing the capability of jumping out of local minimizers, while preserving termination and convergence properties.

Let $\mathbf{x} \in \mathbb{R}^n$ be an n-dimensional vector; $\lfloor \mathbf{x} \rfloor$ denotes the n-dimensional vector whose components are the greatest integers less than or equal to the components of $\mathbf{x}$. The fractional part of $\mathbf{x}$ is denoted as $\mathrm{frac}(\mathbf{x})$ and is defined as follows:

$$\mathrm{frac}(\mathbf{x}) = \mathbf{x} - \lfloor \mathbf{x} \rfloor \tag{7}$$

Let $\boldsymbol{\xi}$ be an arbitrary n-dimensional vector of real numbers and P a positive integer. Let us introduce the following map:

$$\mathbf{x}(\boldsymbol{\xi}) = \mathrm{frac}(P\boldsymbol{\xi}) \tag{8}$$

This map is onto and transforms the n-dimensional Euclidean space $\mathbb{R}^n$ in the n-dimensional unit interval $(0,1)^n$. Let f be the cost function of the minimization problem given in Equation (1) and introduce the following function transformation:

$$\phi(\boldsymbol{\xi}) := f(\mathbf{x}(\boldsymbol{\xi})) = f(\mathrm{frac}(P\boldsymbol{\xi})) \tag{9}$$

where P is a positive integer. Since the cost function f is not necessarily defined outside of the open unit n-box, the domain of the transformed cost function ϕ is:

$$\mathrm{dom}\ \phi = \{\boldsymbol{\xi} \in \mathbb{R}^n : P\boldsymbol{\xi} \notin \mathbb{Z}^n\} \tag{10}$$

The reason is that if $P\boldsymbol{\xi} \in \mathbb{Z}^n$, then $\mathrm{frac}(P\boldsymbol{\xi}) = 0$, and f is not necessarily defined for $\boldsymbol{\xi} = 0$.

The transformed function ϕ has certain interesting properties. The most important are the periodicity of period P^{-1} and piecewise continuity, both on each variable.

Lemma 3. *The transformed function $\phi(\boldsymbol{\xi}) = f(\mathrm{frac}(P\boldsymbol{\xi}))$ satisfies the following properties:*

(i) *The function ϕ is continuous for any $\boldsymbol{\xi} \in \mathbb{R}^n$, such that $P\boldsymbol{\xi} \in (0,1)^n$.*
(ii) *Let $\boldsymbol{\xi}, \boldsymbol{\eta} \in \mathbb{R}^n$ be such that $P\boldsymbol{\xi} \in (0,1)^n$ and $P\boldsymbol{\eta} \in \mathbb{Z}^n$, then $\phi(\boldsymbol{\xi} + \boldsymbol{\eta}) = \phi(\boldsymbol{\xi})$.*

Proof. (i) If $P\boldsymbol{\xi} \in (0,1)^n$, then $\mathrm{frac}(P\boldsymbol{\xi}) = P\boldsymbol{\xi}$ and $\phi(\boldsymbol{\xi}) = f(P\boldsymbol{\xi})$. Since f is continuous for any $\mathbf{x} \in (0,1)^n$, then ϕ is continuous for any $\boldsymbol{\xi} \in \mathbb{R}^n$, such that $P\boldsymbol{\xi} \in (0,1)^n$; (ii) If $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are n-dimensional vectors, such that $P\boldsymbol{\xi} \in (0,1)^n$ and $P\boldsymbol{\eta} \in \mathbb{Z}^n$, respectively, then $\mathrm{frac}(P\boldsymbol{\eta}) = 0$ and $\mathrm{frac}(P(\boldsymbol{\xi} + \boldsymbol{\eta})) = \mathrm{frac}(P\boldsymbol{\xi}) = P\boldsymbol{\xi}$; consequently, $\phi(\boldsymbol{\xi} + \boldsymbol{\eta}) = f(\mathrm{frac}(P\boldsymbol{\xi})) = \phi(\boldsymbol{\xi})$. $\square$

Let $\mathbf{x} \in \mathbb{R}^n$ and $\mathcal{B}_P(\mathbf{x})$ denote the following n-box:

$$\mathcal{B}_P(\mathbf{x}) = \{\boldsymbol{\xi} \in \mathbb{R}^n : \lfloor P\boldsymbol{\xi} \rfloor = \lfloor P\mathbf{x} \rfloor,\ P\boldsymbol{\xi} \notin \mathbb{Z}^n\} \tag{11}$$

The transformed function ϕ is continuous on $\mathcal{B}_P(\mathbf{x})$, and there always exists $\boldsymbol{\xi}^* \in \mathcal{B}_P(\mathbf{x})$, such that $\phi(\boldsymbol{\xi}^*) = \min\{\phi(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \mathrm{dom}\ \phi\}$. Note that the n-box $\mathcal{B}(\mathbf{x})$ is given by $\mathcal{B}_P(\mathbf{x}) = \{\boldsymbol{\xi} = P^{-1}(\mathbf{z} + \lfloor P\mathbf{x} \rfloor) : \mathbf{z} \in (0,1)^n\}$. If $\mathbf{x} = 0$, then $\mathcal{B}_P(\mathbf{0})$ is the fundamental n-box of the function ϕ, *i.e.*, $B_P(\mathbf{0}) = (0, P^{-1})^n$. For another $\mathbf{x} \in \mathbb{R}^n$, $\mathcal{B}_P(\mathbf{x})$ is an n-box of length P^{-1}, where the function ϕ takes the same values as in the fundamental n-box $\mathcal{B}_P(\mathbf{0})$. Note that there are P^n different n-boxes for $\mathbf{x} \in (0,1)^n$; then, the function ϕ repeats P^n times in the unit n-box $(0,1)^n$.

Example 1. The function $f(x) = |x| - |\sqrt{x}\sin(3\pi x)|$, and the transformed function $\phi(\xi) = f(\mathrm{frac}(P\xi))$ for $P = 10$ are depicted for the open unit interval in Figure 1. The function f is continuous on the open unit interval $(0,1)$ and has three local minima. The transformed function ϕ is periodic, of period $P^{-1} = 0.1$, and piecewise continuous on the open unit interval. The fundamental period of ϕ is $\mathcal{B}(0) = (0, 0.1)$, and the function value repeats ten times on the unit interval. The function ϕ is continuous on each interval $\mathcal{B}(x) = \{\xi : \lfloor 10\xi \rfloor = \lfloor 10x \rfloor,\ 10\xi \notin \mathbb{Z}\}$ for $x \in \mathbb{R}$, but discontinuous at the boundary points of these intervals, which are given by $\xi = 0.1k$ for any integer k. This discontinuity always occurs if $f(0 + \epsilon) \neq f(1 - \epsilon)$ for any $\epsilon > 0$, as occurs in this example and which is clearly visible in Figure 1. Moreover, the function ϕ is not necessarily defined for these points of discontinuity.

Consider the unconstrained optimization problem:

$$\min\{\phi(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \mathrm{dom}\ \phi\} \tag{12}$$

The function ϕ has a global optimizer at $\boldsymbol{\xi}^*$ if $\phi(\boldsymbol{\xi}^*) \leq \phi(\boldsymbol{\xi})$ for all $\boldsymbol{\xi} \in \mathrm{dom}\ \phi$. The following theorem proves that the transformed unconstrained minimization problem Equation (12) also provides the minimum value of the original optimization problem Equation (1).

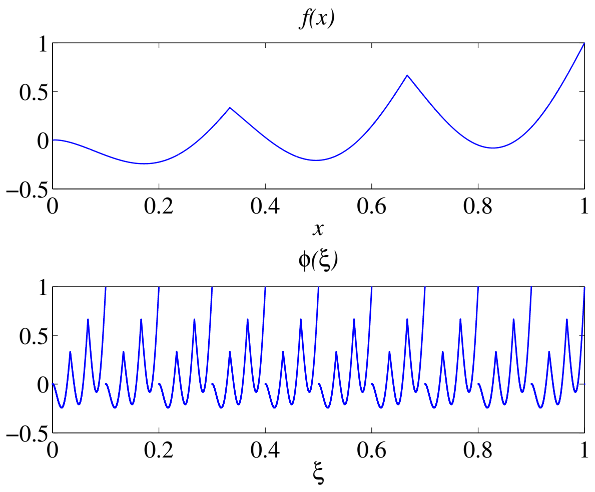

Figure 1. A continuous function $f(x) = |x| - |\sqrt{x}\sin(3\pi x)|$ with three local minima in the unit interval $(0, 1)$ and the corresponding transformed function $\phi(\xi) := f(\text{frac}(P\xi))$ where $P = 10$ for the same interval.

Theorem 1. *Let f be a real value continuous function on the open unit n-box, and let ϕ be defined as $\phi(\xi) = f(\text{frac}(P\xi))$, where P is a positive integer, then $\lambda^* = \min\{f(\mathbf{x}) : \mathbf{x} \in (0, 1)^n\}$ if and only if $\lambda^* = \min\{\phi(\xi) : \xi \in \text{dom}\,\phi\}$.*

Proof. (If): Let $\mathbf{x}^*$ be such that $\lambda^* = f(\mathbf{x}^*) = \min\{f(\mathbf{x}) : \mathbf{x} \in (0, 1)^n\}$ and $\phi(\xi) = f(\text{frac}(P\xi))$, where P is a positive integer; then define $\xi^* = P^{-1}(\mathbf{x}^* + \boldsymbol{\eta})$ for any $\boldsymbol{\eta} \in \mathbb{Z}^n$; then $\phi(\xi^*) = f(\mathbf{x}^*) = \lambda^*$. (Only if): Let ξ^* be such that $\lambda^* = \phi(\xi^*) = \min\{\phi(\xi) : \xi \in \text{dom}\,\phi\}$; define $\mathbf{x}^* = \text{frac}(P\xi^*)$; then $f(\mathbf{x}^*) = \phi(\xi^*) = \lambda^*$. □

Theorem 1 allows us to obtain a global minimizer of the original function f on the open unit n-box by solving the unconstrained optimization problem (12). If we have a method that computes a global minimizer ξ^* of the unconstrained optimization problem (12), then $\mathbf{x}^* = \text{frac}(P\xi^*) \in (0, 1)^n$, and it satisfies $\phi(\xi^*) = f(\mathbf{x}^*)$.

The following lemma proves that any regular n-box of a size greater than or equal to P^{-1} always contains a point ξ^* that attains the global minimum of the unconstrained optimization problem (12).

Lemma 4. *Let $\mathcal{B}$ be a regular n-box of edge length Δ_B, such that $P\Delta_B \geq 1$, and let $\lambda^* = \min\{f(\mathbf{x}) : \mathbf{x} \in (0, 1)\}$; then, the n-box $\mathcal{B}$ always contains a point $\xi^* \in \mathbb{R}^n$, such that $\phi(\xi^*) = \lambda^*$.*

Proof. Let $\mathbf{x}^* \in (0, 1)^n$ be such that $f(\mathbf{x}^*) = \lambda^* \leq f(\mathbf{x})$ for any $\mathbf{x} \in (0, 1)^n$, and let $\bar{\xi} \in \mathcal{B}$, such that $\lfloor P\bar{\xi} \rfloor = \lfloor P\bar{\xi} + \mathbf{x}^* \rfloor$. Consider the open n-box $\mathcal{B}_P(\bar{\xi}) = \{\xi \in \mathbb{R}^n : \lfloor P\xi \rfloor = \lfloor P\bar{\xi} \rfloor,\ P\xi \notin \mathbb{Z}^n\}$, then $\mathcal{B}_P(\bar{\xi})$ has edge length P^{-1}, and the point $\xi^* = P^{-1}(\mathbf{x}^* + \lfloor P\bar{\xi} \rfloor)$ satisfies $\xi^* \in \mathcal{B}(\bar{\xi}) \cap \mathcal{B}$ and $\phi(\xi^*) = f(\mathbf{x}^*) = \lambda^*$. □

We are also interested in determining the size of an n-simplex similar to the standard n-simplex that always contains a global minimizer of $\{\phi(\xi) : \xi \in \text{dom}\,\phi\}$. Since Lemma 4 establishes that such a point is always contained in a regular n-box of length not less than P^{-1}, then the solution is an isosceles right n-simplex that contains a regular n-box of length P^{-1}. From Lemma 1, this n-simplex has edge length nP^{-1} and content $P^{-n}/(n-1)!$. The following lemma states the result.

Theorem 2. *Let Σ be an isosceles right n-simplex of length Δ, such that $P\Delta \geq n$, and let $\lambda^* = \min\{f(\mathbf{x}) : \mathbf{x} \in (0, 1)^n\}$; then, the n-simplex Σ always contains a point $\xi^* \in \text{dom}\,\phi$, such that $f(\text{frac}(\xi^*)) = \lambda^*$.*

Proof. A regular n-box of edge length $\Delta_B = \Delta/n$ can be inscribed inside any isosceles right n-simplex of length Δ. Since $P\Delta \geq n$, then $P\Delta_B \geq 1$, and the theorem statement is a straightforward consequence of Lemma 4. □

In the rest of this section, we shall design a direct search algorithm to obtain a solution for the optimization problem (1). We distinguish two algorithms, the basic algorithm and the complete algorithm. The basic algorithm is a simplex-based algorithm for the transformed cost function. An initial point is embedded at the right vertex of an isosceles right n-simplex of edge length Δ_0. This n-simplex evolves by a sequence of operations. As a result of each iteration, another n-simplex is obtained that is similar to the standard n-simplex and has the point with minimum value of the transformed cost function located in the right vertex. The edge length of the n-simplex decreases by a constant factor if no operations performed during the iteration produce a vertex with a smaller value of the transformed cost function. Whenever the edge length is no less than nP^{-1}, we say that the algorithm is in exploratory phase, because the n-simplex always contains a global minimizer of $\{\phi(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \operatorname{dom} \phi\}$. When the edge length of the n-simplex is less than nP^{-1}, then we say that the algorithm is in convergence phase, and it aims to approach a stationary point. Thus, the basic algorithm is designed to preserve the good properties of direct search algorithms, such as convergence to a stationary point, but increases the possibility to reach the global optimum at the end of the exploratory phase. Another feature of the basic algorithm is that there is no interruption or change from one phase to the other, it is a straightforward consequence of using the transformed cost function.

3.4. *The Basic Algorithm*

The basic algorithm is a simplex-based algorithm for the transformed optimization problem Equation (12). It performs a sequence of operations over an initial simplex until a termination condition is satisfied. The main operators of this algorithm are expansive translation, rotation and shrinkage. The expansive translation is performed after a sufficient decrease of the transformed cost function to ensure that the best point is located at the right vertex. The rotation operation provides exploration of the search space, and the shrinkage operator guarantees termination and global convergence to a stationary point of the transformed cost function. The global improvement is a consequence of using the transformed cost function Equation (8). A sufficient decrease condition of the cost function to accept a successful iteration is a key aspect to prove the termination and global convergence to a stationary point. The main components of the basic algorithm are explained below.

3.4.1. Function Transformation

The cost function transformation is given by:

$$\phi(\boldsymbol{\xi}) = f(\operatorname{frac}(P\boldsymbol{\xi})) \tag{13}$$

This transformation aims to perform global improvement at each iteration $k \in \mathbb{Z}_{[0,\infty)}$ whenever the n-simplex has an edge length, such that $P\Delta_k \geq n$.

3.4.2. Initial Simplex

Let $\operatorname{vset}(\boldsymbol{\xi}, \Delta)$ where $\xi \in \mathbb{R}^n$ and $\Delta \in \mathbb{R}$ be defined as follows:

$$\operatorname{vset}(\boldsymbol{\xi}, \Delta) := \{\boldsymbol{\xi}_i : \boldsymbol{\xi}_i = \boldsymbol{\xi} + \Delta \mathbf{b}_i, i \in \mathbb{Z}_{[1,n]}\} \tag{14}$$

where $\mathbf{b}_0$ is an n-dimensional zero vector and $\mathbf{b}_i$ for $i \in \mathbb{Z}_{[1,n]}$ is the i-th element of the standard basis of $\mathbb{R}^n$. The initial n-simplex is obtained by varying an initial point at a distance Δ_0 in each coordinate direction, such that $P\Delta_0 > n$. Let $\boldsymbol{\xi}_0 \in (0,1)^n$ be an initial point; then, the vertex set of the initial simplex is given by:

$$\mathcal{V}_0 = \operatorname{vset}(\boldsymbol{\xi}_0, \Delta_0) \tag{15}$$

If the initial point $\boldsymbol{\xi}_0$ is not provided as an input of the algorithm, then it can be randomly chosen. Consequently, the initial simplex is an isosceles right n-simplex with edge length Δ_0 and content $\frac{1}{n!}\Delta_0^n$ whose right vertex is indexed by zero.

3.4.3. Expansive Translation

This operation aims to position the point with lowest value of the transformed cost function found so far at the right vertex. Let $\mathcal{V}$ be the vertex set of an isosceles right n-simplex with edge length Δ, and let $\boldsymbol{\xi}_i \in \mathcal{V}$ be such that $\phi(\boldsymbol{\xi}_i) \leq \phi(\boldsymbol{\xi}_j)$ for $\boldsymbol{\xi}_j \in \mathcal{V}$; then, the expansive translation of the right n-simplex $\mathcal{V}$ with respect to the vertex i is given by the following expression:

$$\boldsymbol{\xi}'_j = \boldsymbol{\xi}_i + \rho(\boldsymbol{\xi}_j - \boldsymbol{\xi}_0),\ \boldsymbol{\xi}_j \in \mathcal{V} \tag{16}$$

where $1 \leq \rho < \infty$. If the vertex set of the current simplex is given by $\mathcal{V} = \text{vset}(\boldsymbol{\xi}_0, \Delta)$, then the transformed n-simplex has the vertex set $\mathcal{V}' = \text{vset}(\boldsymbol{\xi}_j, \Delta')$, where $\Delta' = \rho\Delta$, and its edge length is $|\Delta'|$. The new n-simplex, after the expansive translation, preserves its shape, but increases its content by a factor of ρ^n. From a practical point of view, it is convenient to choose the expansive factor ρ close to one, but slightly greater, because this avoids obtaining candidate points that were already visited in previous iterations, in the next rotation operation.

Let c be an arbitrary positive constant; if $\phi(\boldsymbol{\xi}'_i) < \phi(\boldsymbol{\xi}_0) - c\Delta^2$ for some $\boldsymbol{\xi}'_i \in \mathcal{V}'\}$, then the algorithm proceeds to a new iteration and performs an expansive translation operation. Otherwise, the algorithm executes a rotation operation.

The introduction of the positive term $c\Delta^2$ guarantees that the iteration is considered successful only if it provides a point with a sufficient decrease of the cost function.

3.4.4. Rotation

This operation produces a similar n-simplex, but where each edge is rotated π radians with respect to the right vertex. A graphical interpretation is depicted in Figure 2.

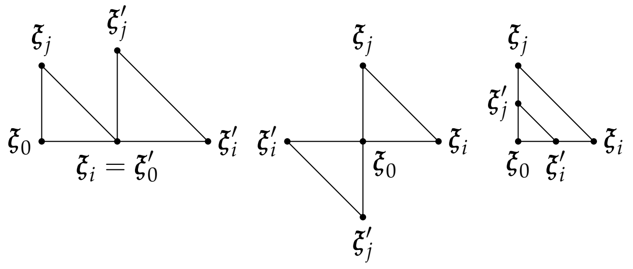

Figure 2. Graphical representation of the simplex translation, rotation and shrinking simplex operations for three points in $\mathbb{R}^2$. The initial vertex set is $\mathcal{V} = \{\boldsymbol{\xi}_0, \boldsymbol{\xi}_i, \boldsymbol{\xi}_j\}$, and the final vertex set is $\mathcal{V}' = \{\boldsymbol{\xi}'_0, \boldsymbol{\xi}'_i, \boldsymbol{\xi}'_j\}$. The three operations produce similar simplices.

Let $\mathcal{V}$ denote the vertex set of an n-simplex. The vertices of the rotated n simplex are given by the following expression:

$$\boldsymbol{\xi}'_j = \boldsymbol{\xi}_0 - (\boldsymbol{\xi}_j - \boldsymbol{\xi}_0),\ \ \boldsymbol{\xi}_j \in \mathcal{V} \tag{17}$$

If the vertex set of the current simplex is given by $\mathcal{V} = \text{vset}(\boldsymbol{\xi}_0, \Delta)$, then the transformed n-simplex has the vertex set $\mathcal{V}' = \text{vset}(\boldsymbol{\xi}_0, \Delta')$, where $\Delta' = -\Delta$, and its edge length is $|\Delta'|$. If $\phi(\boldsymbol{\xi}'_i) < \phi(\boldsymbol{\xi}_0) - c\Delta^2$ for some $\boldsymbol{\xi}'_i \in \mathcal{V}'$, then the rotation operation succeeds, and the algorithm proceeds to start a new iteration and performs an expansive translation operation. Otherwise, the rotation operation fails; the rotated vertex set is discarded; and the algorithm executes a shrinkage operation with the original vertex set $\mathcal{V}$.

3.4.5. Shrinkage

This operation performs a contraction of the simplex by fixing the right vertex and placing the remaining vertices along the same directions, but at a distance that is reduced by a constant factor $\sigma \in (0, 1)$. A graphical interpretation of the shrinkage operation is depicted in Figure 2 for $\mathbb{R}^2$.

Let $\mathcal{V}$ denote the vertex set of an n-simplex. The vertices of the transformed n-simplex are given by:

$$\boldsymbol{\xi}'_j = \boldsymbol{\xi}_0 + \sigma(\boldsymbol{\xi}_j - \boldsymbol{\xi}_0),\ \boldsymbol{\xi}_j \in \mathcal{V} \tag{18}$$

If the vertex set of the current simplex is given by $\mathcal{V} = \text{vset}(\boldsymbol{\xi}_0, \Delta)$, then the transformed n-simplex has the vertex set $\mathcal{V}' = \text{vset}(\boldsymbol{\xi}_0, \Delta')$, where $\Delta' = \sigma\Delta$, and its edge length is $|\Delta'|$. After the shrinkage operation, the algorithm proceeds to start a new iteration by performing an expansive translation operation.

3.4.6. Sufficient Decrease Condition

The expansive translation and rotation operations are considered successful if some vertex of the transformed n-simplex satisfies the following sufficient decrease condition:

$$\phi(\boldsymbol{\xi}'_i) < \phi(\boldsymbol{\xi}_0) - c\Delta^2, \text{ for some } \boldsymbol{\xi}'_i \in \mathcal{V}'$$

where c is a positive constant. This condition is key to proving the termination of the basic algorithm and global convergence to a stationary point. The constant c is arbitrary, but usually, a small value is chosen.

3.4.7. Stopping Criterion

The sufficient decrease condition and the shrinkage operation guarantee that the size of the n-simplex converges to zero. Consequently, a stopping criterion can be defined by taking a sufficiently small number $\epsilon > 0$. Let $|\Delta|$ be the edge length of the n-simplex; then, the basic algorithm stops when the following condition is satisfied:

$$P|\Delta| \leq \epsilon \tag{19}$$

3.4.8. Space Transformation

The search space transformation is given by the map:

$$x(\boldsymbol{\xi}) = \text{frac}(P\boldsymbol{\xi}) \tag{20}$$

Therefore, when the n-simplex with vertex set $\mathcal{V}$ reaches edge length $\Delta \leq \epsilon$, the basic algorithm stops, and the point obtained is $\bar{\mathbf{x}} = \text{frac}(P\bar{\boldsymbol{\xi}})$, where $\bar{\boldsymbol{\xi}} \in \arg\min\{\boldsymbol{\xi} : \boldsymbol{\xi} \in \mathcal{V}\}$; and it attains the function value $f(\bar{\mathbf{x}}) = \phi(\bar{\boldsymbol{\xi}})$.

3.4.9. Implementation

An implementation of the basic algorithm in pseudocode is given in Algorithm 1.

Algorithm 1 Basic algorithm.

```
function BGDS($f, \mathbf{x}_0$)
    parameters: $\epsilon > 0, P \in \mathbb{N}$
    definitions: $s(\boldsymbol{\xi}) := P\boldsymbol{\xi} - \lfloor P\boldsymbol{\xi} \rfloor, \phi(\boldsymbol{\xi}) := f(s(\boldsymbol{\xi}))$
    $\mathbf{z} \leftarrow \mathrm{rand}(\dim(x_0))$
    $\boldsymbol{\xi}_0 \leftarrow (\lfloor P\mathbf{z} \rfloor + \mathbf{x}_0)/P$
    $\epsilon \leftarrow \epsilon/P$
    $\boldsymbol{\xi} \leftarrow \mathrm{LDS}(\phi, \boldsymbol{\xi}_0, \epsilon)$                    ▷ Local direct search
    return $s(\boldsymbol{\xi})$
end function
function LDS($\phi, \boldsymbol{\xi}_0, \epsilon$)
    parameters: $\rho, \sigma, c \in \mathbb{R} : 1 \leq \rho, 0 < \sigma < 1, 0 < c, 0 < \Delta$
    definitions: $\mathrm{vset}(\boldsymbol{\xi}, \Delta) := \boldsymbol{\xi} \in \mathbb{R}^n, \Delta \in \mathbb{R}\}, \mathrm{vmin}(\mathcal{V}) \in \arg\min\{\phi(\boldsymbol{\xi}_i) : \boldsymbol{\xi}_i \in \mathcal{V}\}$
    $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi}_0$
    $\mathcal{V} \leftarrow \mathrm{vset}(\boldsymbol{\xi}, \Delta)$
    $\boldsymbol{\xi}_{\min} \leftarrow \mathrm{vmin}(\mathcal{V})$
    while $\epsilon < |\Delta|$ do
        if $\phi(\boldsymbol{\xi}_{\min}) < \phi(\boldsymbol{\xi}) - c\Delta^2$ then          ▷ Expansive translation
            $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi}_{\min}$
            $\Delta \leftarrow \rho\Delta$
            $\mathcal{V} \leftarrow \mathrm{vset}(\boldsymbol{x}, \Delta)$
            $\boldsymbol{\xi}_{\min} \leftarrow \mathrm{vmin}(\mathcal{V})$
        end if
        if $\phi(\boldsymbol{\xi}) - c\Delta^2 \leq \phi(\boldsymbol{\xi}_{\min})$ then          ▷ Rotation
            $\Delta \leftarrow -\Delta$
            $\mathcal{V} \leftarrow \mathrm{vset}(\boldsymbol{\xi}, \Delta)$
            $\boldsymbol{\xi}_{\min} \leftarrow \mathrm{vmin}(\mathcal{V})$
        end if
        if $\phi(\boldsymbol{\xi}) - c\Delta^2 \leq \phi(\boldsymbol{\xi}_{\min})$ then          ▷ Shrinkage
            $\Delta \leftarrow \sigma\Delta$
            $\mathcal{V} \leftarrow \mathrm{vset}(\boldsymbol{\xi}, \Delta)$
            $\boldsymbol{\xi}_{\min} \leftarrow \mathrm{vmin}(\mathcal{V})$
        end if
    end while
    return $\boldsymbol{\xi}_{\min}$
end function
```

3.5. Convergence of the Basic Algorithm

The basic algorithm has been designed to provide termination and improvement by performing a sequence of simplex transformations. If in addition, the cost function is differentiable and its gradient is Lipschitz continuous, then the basic algorithm converges to a stationary point.

Theorem 3. *Let f be the cost function of a minimization problem on the open unit n-box; then the basic algorithm always terminates. If, in addition, f is differentiable and its gradient Δf is Lipschitz continuous on the unit n-box, then the sequence of points generated at each iteration by the basic algorithm converges to a stationary point.*

Proof. Let $\mathcal{S}$ and $\mathcal{U}$ denote the index sets of successful and unsuccessful iterations of the basic algorithm, respectively. Let $\mathbf{x}_k = \mathrm{frac}(P\boldsymbol{\xi}_k)$, then $f(\mathbf{x}_k) = \phi(\boldsymbol{\xi}_k)$. If $k \in \mathcal{S}$, then $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k) - c\Delta_k^2$ and $\Delta_{k+1} = \rho\Delta_k$, while if $k \in \mathcal{U}$, then $f(\mathbf{x}_{k+1}) \geq f(\mathbf{x}_k) - c\Delta_k^2$ and $\Delta_{k+1} = \sigma\Delta_k$. Let us begin by proving that $\lim_{k\to\infty} |\Delta_k| = 0$. Suppose not; then, for any $k \in \mathbb{Z}_{[0,\infty]}$, there exists $\ell \geq k$, such that $\ell \in \mathcal{S}$ because, if such an ℓ does not exist, then the set of successful iterations would be finite, and when k approaches infinity, the update rule $\Delta_{k+1} = \sigma\Delta_k$ yields $\lim_{k\to\infty}|\Delta_k| = 0$. In addition, there exists $\underline{\Delta} > 0$, such that for each successful iteration $\ell \in \mathcal{S}$, $|\Delta_\ell| \geq \underline{\Delta}$ and:

$$f(\mathbf{x}_{\ell+1}) < f(\mathbf{x}_\ell) - c\Delta_\ell^2 \leq f(\mathbf{x}_\ell) - c\underline{\Delta}^2$$

However, since ℓ approaches infinity, this implies that $f(\mathbf{x}_{\ell+1}) \to -\infty$, which contradicts the fact that the function f is bounded below on $(0,1)^n$. Consequently, the boundedness of f implies that $\underline{\Delta} = 0$ and $\lim_{k\to\infty}|\Delta_k| = 0$. This proves the termination of the basic algorithm.

For the second part of the theorem, if f is continuous and differentiable on $(0,1)^n$, then so is $\phi(\boldsymbol{\xi}) = f(\mathrm{frac}(P\boldsymbol{\xi}))$ on the n-box $\mathcal{B}_P(\mathbf{x}) = \{\boldsymbol{\xi} \in \mathbb{R}^n : \lfloor P\boldsymbol{\xi}\rfloor = \lfloor P\mathbf{x}\rfloor,\ P\boldsymbol{\xi} \notin \mathbb{Z}^n\}$ for any $\mathbf{x} \in \mathbb{R}^n$. Furthermore, if ∇f is Lipschitz continuous with constant M on $(0,1)^n$, then so is $\nabla\phi$ in $\mathcal{B}_P(\mathbf{x})$ with constant PM. Since we proved that $\lim_{k\to\infty}|\Delta_k| = 0$, then there always exists a positive integer L, such that $\lfloor P\boldsymbol{\xi}_k\rfloor = \lfloor P\boldsymbol{\xi}_L\rfloor$ for any positive integer $k > L$. The basic algorithm searches points along the coordinate directions; then, from Lemma 2 and for any positive integer $k > L$, no matter the value of $\nabla\phi(\boldsymbol{\xi}_k)$, there is always at least one (positive or negative) coordinate direction $\mathbf{d}_i \in \{\mathbf{b}_i, -\mathbf{b}_i\}$, such that:

$$\frac{1}{\sqrt{n}} \leq \cos\theta = \frac{\langle -\nabla\phi(\boldsymbol{\xi}_k), \mathbf{b}_i\rangle}{\|\nabla\phi(\boldsymbol{\xi}_k)\|\|\mathbf{d}_i\|}$$

or equivalently:

$$\frac{1}{\sqrt{n}}\|\nabla\phi(\boldsymbol{\xi}_k) \leq -\langle\nabla\phi(\boldsymbol{\xi}_k), \mathbf{d}_i\rangle \tag{21}$$

because $\|\mathbf{d}_i\| = 1$. For any unsuccessful iteration $k \in \mathcal{U}$, $\phi(\boldsymbol{\xi}_k + |\Delta_k|\mathbf{d}_k) \geq \phi(\boldsymbol{\xi}_k) - c\Delta_k^2$, the mean value theorem establishes that:

$$\begin{aligned}-c\Delta_k^2 &\leq \phi(\boldsymbol{\xi}_k + |\Delta_k|\mathbf{d}_i) - \phi(\boldsymbol{\xi}_k)\\ &= \langle\nabla\phi(\boldsymbol{\xi}_k + \alpha_k|\Delta_k|\mathbf{d}_i), \mathbf{d}_i\rangle|\Delta_k|\end{aligned}$$

for some $\alpha_k \in [0,1]$. Subtracting $\langle\nabla\phi(\boldsymbol{\xi}_k), \mathbf{d}_i\rangle|\Delta_k|$ from both sides leads to:

$$-c\Delta_k^2 - \langle\nabla\phi(\boldsymbol{\xi}_k), \mathbf{d}_i\rangle|\Delta_k| \leq \langle\nabla\phi(\boldsymbol{\xi}_k + \alpha_k|\Delta_k|\mathbf{d}_i) - \nabla\phi(\boldsymbol{\xi}_k), \mathbf{d}_i\rangle|\Delta_k|$$

Applying Equation (21) yields:

$$\frac{1}{\sqrt{n}}\|\nabla\phi(\boldsymbol{\xi}_k)\| \leq \langle\nabla\phi(\boldsymbol{\xi}_k + \alpha_k|\Delta_k|\mathbf{d}_i) - \nabla\phi(\boldsymbol{\xi}_k), \mathbf{d}_i\rangle + c|\Delta_k|$$

Taking into account that ϕ is continuously differentiable in $\mathcal{B}_P(\boldsymbol{\xi}_k)$ and its gradient $\nabla\phi$ is Lipschitz with constant PM in $\mathcal{B}_P(\boldsymbol{\xi}_k)$,

$$\frac{1}{\sqrt{n}}\|\nabla\phi(\boldsymbol{\xi}_k)\| \leq PM\|\alpha_k\Delta_k\mathbf{d}_i\| + c|\Delta_k| \leq (PM + c)|\Delta_k|$$

because $\alpha_k \in (0,1)$ and $\|\mathbf{b}_i\| = 1$. Consequently,

$$\|\nabla\phi(\boldsymbol{\xi}_k)\| \leq \sqrt{n}(PM + c)|\Delta_k|$$

and taking the limit when k approaches infinity, it is possible to conclude that:

$$\lim_{k\to\infty}\|\nabla\phi(\boldsymbol{\xi}_k)\| \leq \sqrt{n}(PM + c)\lim_{k\to\infty}|\Delta_k| = 0, \text{ for } k \in \mathcal{U}$$

and finally, by making $\mathbf{x}_k = \mathrm{frac}(\boldsymbol{\xi}_k)$:

$$\lim_{k\to\infty}\|\nabla f(\mathbf{x}_k)\| = P\lim_{k\to\infty}\|\nabla\phi(\boldsymbol{\xi}_k)\| = 0, \text{ for } k \in \mathcal{U}.$$

Consequently, the sequence of points generated by the basic algorithm converges to a stationary point. □

Remark 2. The basic algorithm can be applied to any cost function f, not necessarily continuous or differentiable in the search space $(0,1)^n$. For a continuous and differentiable function with a Lipschitz continuous gradient, the basic algorithm converges to a stationary point. If in addition the cost function is strictly convex, then the unique optimizer is a stationary point, and Theorem 3 guarantees that the basic algorithm converges to the unique optimizer. Note that we do not need to know the value of the Lipschitz constant M.

Example 2. The evolution of the basic algorithm for the function $f(x) = |x| - |\sqrt{x}\sin(3\pi x)|$ is depicted in Figure 3. The parameters of the algorithm are $P = 10^4$, $\sigma = 0.5$, $\rho = 1.05$, $c = 0.01$, $\epsilon = 10^{-4}$. Since this problem is unidimensional, the simplex is an interval. The left plots represent the vertex points of the simplex in the original space (above) and their corresponding function values (below) after each iteration. The right plots represent the values of the worst (above) and best (below) points of the simplex in the original space that were obtained during the execution of the algorithm. Note that the algorithm explores the whole range of the unit interval. This exploration is performed whenever the length of the simplex edge Δ_k satisfies $P\Delta_k \geq 1$, which corresponds to the exploratory phase of the algorithm. Once Δ_k is $P\Delta_k < 1$, then the algorithm enters the phase of convergence to a stationary point. In this example, the algorithm stops after 24 iterations. From Iterations 1 to 12, the algorithm is in the exploratory phase, and from Iterations 13 to 24, the algorithm is in the convergence phase to a stationary point. The point obtained corresponds to $f(0.1721) = -0.2422$, which is the global minimal value within the space tolerance ϵ.

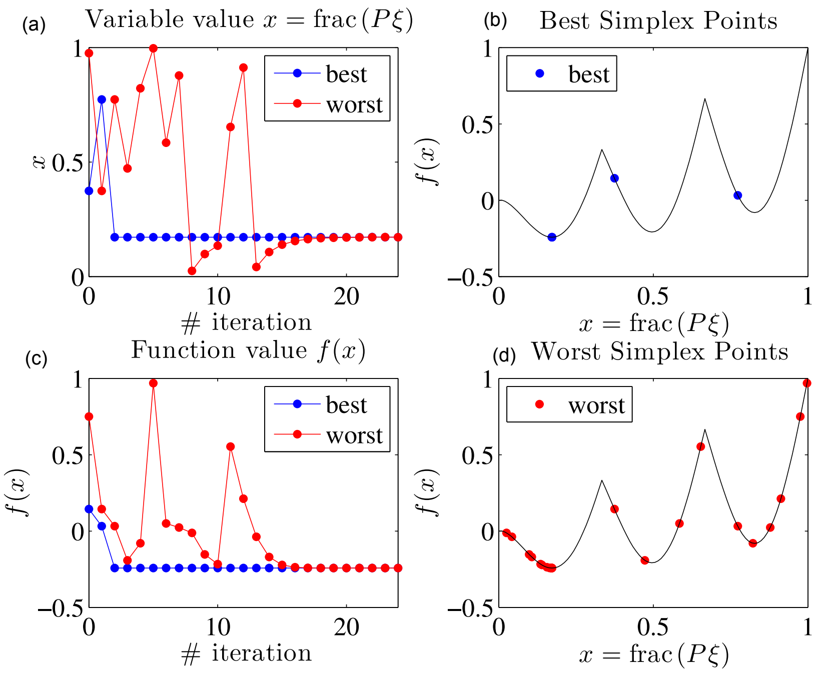

Figure 3. Evolution of the algorithm for $f(x) = |x| - |\sqrt{x}\sin(3\pi x)|$ and $x(\xi) = \mathrm{frac}(P\xi)$. The parameters of the algorithm are $P = 10^4$, $\sigma = 0.5$, $\rho = 1.05$, $c = 0.01$, $\epsilon = 10^{-4}/P$.

3.6. The Complete Algorithm

The complete algorithm, that we call GDS (an acronym for Global Direct Search), is obtained by repeating the basic algorithm a number of times. A single application of the basic algorithm may obtain the global optimum within a space tolerance ϵ only in easy problems. For more difficult optimization problems, a new execution of the basic algorithm, starting with the best point obtained in the previous execution, may improve the result. Each execution produces a local minimizer; however, if there are better points in some coordinate direction, the algorithm may jump to one of these points during the exploratory phase of a new execution. Unfortunately, not every optimization problem satisfies the property that, for any local minimizer, there are points in some coordinate direction where the cost function is improved. In such a case, a strategy that turns out to be more effective is to start from a new random initial point. The class of cost functions that we are facing is not usually known in advance, and we do not know which strategy is more appropriate for a new execution of the basic algorithm. A trade-off solution is to repeat the basic algorithm N times, but starting from a new random initial point after a number $R \leq N$ of repetitions.

3.6.1. Initial Point

Let N denote the total number of executions of the basic algorithm and R the number of repetitions without choosing a new initial point. Let $\bar{\mathbf{x}}$ denote the point obtained in the previous execution of the basic algorithm and $\boldsymbol{\xi}_r$ a random vector in $(0,1)^n$ with a uniform probability distribution. The initial point for the n-th execution of the basic algorithm is obtained as follows:

$$\boldsymbol{\xi}_0 = \begin{cases} \operatorname{frac}(P^{-1}\operatorname{int}(P\boldsymbol{\xi}_r) + \bar{\mathbf{x}}) & \text{if } n \bmod (R+1) \neq 0 \\ \boldsymbol{\xi}_r & \text{if } n \bmod (R+1) = 0 \end{cases} \tag{22}$$

where mod is the modulo operator that represents the remainder of the integer division of their arguments. Note that if $R = 0$, the initial point is always a random point; however, if $R = N$, the initial point is always the point obtained in the previous execution of the basic algorithm.

3.6.2. Best Point Storage

The minimum point found during the execution of the complete algorithm is preserved. The point obtained after each basic algorithm execution is compared to the best solution stored and, in the case of improvement, the new point replaces it. Let $\mathbf{x}$ denote the point obtained after the last execution of the basic algorithm and $\mathbf{x}_{\min}$ the best point stored so far; the storage rule is:

$$\mathbf{x}'_{\min} = \begin{cases} \bar{\mathbf{x}} & \text{if } f(\bar{\mathbf{x}}) < f(\mathbf{x}_{\min}) \\ \mathbf{x}_{\min} & \text{otherwise} \end{cases} \tag{23}$$

3.6.3. Implementation of the Complete Algorithm

An implementation of the complete algorithm GDS in pseudocode is given in Algorithm 2.

Algorithm 2 Complete algorithm GDS.

```
1:  function GDS($f, \mathbf{x}_0$)
2:      parameters: $N, R \in \mathbb{N}$, such that $R \leq N$
3:      $\mathbf{x}_{\min} \leftarrow \mathbf{x}_0$
4:      for $k = 1$ to $N$ do
5:          $\mathbf{x} \leftarrow \mathbf{x}_{\min}$
6:          if $k \bmod (R+1) = 0$ then
7:              $\mathbf{x} \leftarrow \text{rand}(\dim(\mathbf{x}_0))$
8:          end if
9:          $\mathbf{x} \leftarrow \text{BGDS}(f, \mathbf{x})$          ▷ Basic global direct search
10:         if $f(\mathbf{x}) < f(\mathbf{x}_{\min})$ then
11:             $\mathbf{x}_{\min} \leftarrow \mathbf{x}$
12:         end if
13:     end for
14:     return $\mathbf{x}_{\min}$
15: end function
```

4. Experimental Study

4.1. Test Functions

Global optimization is not a trivial problem. Theoretical results, such as the No Free Lunch (NFL) theorem [61], state that an efficient general-purpose universal optimization algorithm is not possible. In other words, this result says that one algorithm can outperform another only for a certain class of problems. It is a usual practice to analyze the performance of an optimization algorithm by using test functions. A large number of standard test functions have been used to study the performance of our direct search global optimization algorithm. In this section, we present the results for the 24 functions given in Table 1. These function are described in detail in [62]. The functions are classified into five groups: separable functions (func # 1.xx), functions with low or moderate conditioning (func # 2.xx), unimodal functions with high conditioning (func # 3.xx), multimodal functions with an adequate global structure (func # 4.xx) and multimodal functions with a weak global structure (func # 5.xx). All of the functions are scalable with the dimension. All of the functions are defined and can be evaluated over $\mathbb{R}^n$, and the search space is $[-5, 5]^n$, where n is the dimension of the function domain. The global optimal value is attained in this search space.

Table 1. Description of the functions used in the experimental study.

Func #	Description
1.01	Sphere
1.02	Ellipsoid separable with monotone x-transformation, condition 1e + 06
1.03	Rastrigin separable with both x-transformations condition 10
1.04	Skew Rastrigin–Bueche separable, condition 10, skew-condition 100
1.05	Linear slope, neutral extension outside the domain (not flat)
2.06	Attractive sector function
2.07	Step-ellipsoid, condition 100
2.08	Rosenbrock, original
2.09	Rosenbrock, rotated
3.10	Ellipsoid with monotone x-transformation, condition 1e6
3.11	Discus with monotone x-transformation, condition 1e6
3.12	Bent cigar with asymmetric x-transformation, condition 1e6
3.13	Sharp ridge, slope 1:100, condition 10
3.14	Sum of different powers
4.15	Rastrigin with both x-transformations, condition 10
4.16	Weierstrass with monotone x-transformation, condition 100
4.17	Schaffer F7 with asymmetric x-transformation, condition 10
4.18	Schaffer F7 with asymmetric x-transformation, condition 1000
4.19	F8F2 composition of 2D Griewank–Rosenbrock
5.20	Schwefel x sin(x) with tridiagonal transformation, condition 10
5.21	Gallagher 101 Gaussian peaks, condition up to 1000
5.22	Gallagher 21 Gaussian peaks, condition up to 1000, 1000 for global opt
5.23	Katsuuras repetitive rugged function
5.24	Lunacek bi-Rastrigin, condition 100

4.2. Selection of the GDS Parameters

The performance of the GDS algorithm for different parameter values has been empirically analyzed by means of extensive computer studies. Our algorithm has been implemented in the scientific software MATLAB [63], and its performance has been successfully compared to the global optimization strategies that are implemented in the Global Optimization Toolbox of MATLAB. The GDS algorithm has a number of configurable parameters, and this comparative study allowed us to select a set of parameters that are appropriate for many problems. The expansive and contractive factors are set to $\rho = 1.05$ and $\sigma = 0.5$. It is convenient to take the expansive value slightly greater than one to avoid evaluating points already checked in previous iterations. However, large values of ρ usually produce slow convergence. Consequently, values in the interval $[1.01, 1.10]$ are usually a good option for most problems. Regarding the contractive factor σ, the closer to one, the larger the number of iterations of the basic GDS algorithm will be. From our experiments, $\sigma = 0.5$ is a convenient value that produces a satisfactory trade-off between the efficiency of the algorithm and the convergence time. The constant c of the sufficient decrease condition is chosen as $c = 0.01$. Any positive value of c provides the termination of the algorithm, but small values are recommended to improve the algorithm's performance during the initial iterations when Δ is large. The parameters P and R were fixed to $P = 1000$ and $R = 5$. These values were selected after a massive experimentation. In our experiments, we analyzed the mean value of the CPU time for one execution of the basic GDS algorithm for different values of the parameter R. The empirical results showed that the larger the value of R, the lower the CPU time for execution. This means that when R increases, the number of executions N can also be increased to a certain extent without compromising the total CPU time. The reduction is larger for smaller values of R, e.g., the reduction is about 40% from $R = 0$ to $R = 1$ and about 30% from $R = 1$ to $R = 2$. Since increasing R is also beneficial for functions that can be globally optimized along the coordinate directions, a good option that trades off between moderate CPU time and efficiency in computing the global optimum is to choose $R \in [2, 10]$ and increase the number of repetitions of the basic algorithm N as much as possible, to maintain a moderate total CPU time. In fact, we have checked that a good selection of P and R could improve the results depending on the problem. However, since a practitioner usually does not know in advance what class its problem belongs to, it is important to fix a set of parameters that trade off for a large class of problems. In conclusion, the parameter values of the GDS algorithm for the experimental study were selected as $P = 1000$, $R = 5$, $\rho = 1.05$, $\sigma = 0.5$, $c = 0.01$ and $N = 1e9$, and the stopping criterion is the number of function evaluations.

4.3. Performance Evaluation

The evaluation of the GDS algorithm has been accomplished by solving the 24 functions in Table 1 using the fixed set of parameters values explained in the previous subsection. The performance of GDS is compared to DIRECT [51]. DIRECT is a deterministic global optimization algorithm based on the domain partition strategy. Each iteration includes two phases. The first phase identifies hyper-rectangles that potentially contain a global optima. In the second phase, the hyper-rectangles identified in the first phase are partitioned into smaller hyper-rectangles, and the cost function is evaluated at the centers of the hyper-rectangles. The convergence properties guarantee that if no termination criteria are included, DIRECT will exhaustively sample the domain. The implementation of DIRECT in MATLAB by Finkel [64] has been used in this study. The experiments have been performed in a laptop PC equipped with a dual core processor Intel (R) Core (TM) i5-3317U CPU @1.70 GHz running MATLAB R2014a under the Windows operating system.

The comparative study has been performed as follows: For both algorithms, GDS and DIRECT, a fixed number of function evaluations is used as a stopping condition, and the error in computing the global minimum value is obtained. Each problem has been solved 15 times, and the median of the error is tabulated for each function. The study has been accomplished for the 24 functions of Table 1 and for different dimensions of the function domain and different numbers of function evaluations. The default parameters of DIRECT [64] have been used, but using the number of function evaluations as stopping conditions.

In Table 2, the results for functions of dimensions five and 10 and a number of function evaluations equal to 10^3 and 10^4 are shown, while in Table 3, the same results are shown for dimensions 10 and 20 and 10^4 and 10^5 function evaluations. GDS and DIRECT obtained different results for different functions. In general, for a small number of function evaluations, DIRECT gets a smaller median of the error. However, as the number of function evaluations increases, the improvement in approaching the minimum value is higher in GDS. One of the outstanding features of GDS is its simplicity. This can be checked by comparing the CPU time with DIRECT. The CPU time for all of the problems, dimensions and function evaluations is much smaller for GDS; see Figures 4 and 5. Note that GDS is more than 100-times faster than DIRECT for several functions of dimension 40 and 10^5 function evaluations.

Table 2. Comparative study of the median of the error for problems of dimensions 5 and 10 and the number of evaluations 1e + 03 and 1e + 04.

Func #	**GDS dim = 5**	**DIRECT nev = 1e+03**	**GDS dim = 5**	**DIRECT nev = 1e+04**	**GDS dim = 10**	**DIRECT nev = 1e+03**	**GDS dim = 10**	**DIRECT nev = 1e+04**
1.01	3.647e+01	5.094e−04	9.920e−09	1.347e−04	1.422e+02	1.862e−01	9.826e−09	1.076e−03
1.02	2.400e+06	2.044e+00	8.064e−06	3.053e−01	1.072e+07	1.475e+05	2.020e−05	2.129e+00
1.03	3.649e+02	7.015e+00	2.002e-02	5.970e+00	6.338e+02	4.234e+01	7.859e−02	1.692e+01
1.04	2.507e+02	1.249e+01	1.696e−02	6.967e+00	9.518e+02	5.252e+01	1.058e+00	2.274e+01
1.05	6.767e+01	1.643e−02	7.809e−05	1.643e-02	1.788e+02	2.271e+01	1.986e−04	1.909e−02
2.06	6.348e+04	2.306e+00	3.497e−07	7.272e−01	5.778e+05	2.520e+01	1.435e−01	1.185e+01
2.07	4.044e+00	6.738e−01	5.543e−01	5.216e−03	2.796e+01	6.227e+00	5.254e+00	1.815e+00
2.08	3.644e+04	2.107e+00	1.647e+00	1.794e−02	2.066e+05	9.353e+01	3.141e+00	8.302e+00
2.09	7.835e+03	5.858e-01	2.433e−01	2.747e−02	8.654e+04	1.551e+01	5.383e+00	6.010e+00
3.10	1.426e+05	8.102e+02	9.892e+02	3.345e+01	7.788e+06	1.312e+04	3.999e+03	1.640e+03
3.11	1.003e+02	1.997e+01	5.722e+01	4.615e+00	1.647e+06	5.343e+01	1.023e+02	3.198e+01
3.12	4.213e+07	6.904e+00	4.372e+00	2.012e−02	2.597e+08	1.284e+05	5.751e+00	7.904e−01
3.13	8.841e+02	2.545e+00	1.067e+00	5.285e−02	2.388e+03	9.999e+01	2.036e+00	6.097e+00
3.14	2.356e+01	8.957e−03	8.020e−04	7.238e−04	5.779e+01	2.685e−01	2.232e−03	1.452e−02
4.15	1.920e+02	7.971e+00	5.014e+00	2.988e+00	7.494e+02	5.364e+01	6.365e+01	1.991e+01
4.16	3.869e+00	6.053e−01	6.144e−01	4.247e−03	3.089e+01	5.140e+00	2.428e+00	5.762e−01
4.17	1.069e+01	1.104e−01	9.849e−01	6.272e−03	2.350e+01	1.457e+00	1.265e+01	9.732e−02
4.18	4.694e+01	2.134e−01	2.395e+00	2.908e−02	5.608e+01	6.410e+00	5.172e+01	4.363e−01
4.19	2.775e+00	8.478e-02	5.901e−01	4.858e-02	2.127e+01	1.990e-01	2.630e+00	1.620e−01
5.20	1.983e+04	1.265e+00	2.392e−01	8.295e−01	7.714e+04	2.707e+00	9.283e-01	1.829e+00
5.21	1.751e+01	2.052e−04	9.300e−01	6.972e−06	7.886e+01	2.338e+00	6.676e+00	2.239e−03
5.22	4.032e+01	6.928e−01	6.024e−02	2.746e−04	8.485e+01	1.651e+00	7.272e+00	6.928e−01
5.23	1.395e+00	9.682e−01	6.205e−01	8.782e−01	2.247e+00	1.438e+00	7.203e−01	9.066e−01
5.24	9.572e+01	2.122e+01	1.108e+01	1.071e+01	2.673e+02	7.556e+01	9.847e+01	6.139e+01

Table 3. Comparative study of the median of the error for problems of dimensions 20 and 40 and the number of evaluations 1e + 04 and 1e + 05.

Func #	**GDS dim = 20**	**DIRECT nev = 1e+04**	**GDS dim = 20**	**DIRECT nev = 1e+05**	**GDS dim = 40**	**DIRECT nev = 1e+04**	**GDS dim = 40**	**DIRECT nev = 1e+05**
1.01	6.690e−2	1.456e−01	9.434e-09	2.626e−03	3.187e+02	2.533e+01	8.897e−09	5.976e−02
1.02	3.171e+03	1.570e+06	3.806e−05	3.471e+01	1.195e+07	2.904e+06	4.865e−02	2.904e+06
1.03	9.586e+00	5.187e+01	3.545e−02	3.620e+01	1.719e+03	5.029e+02	2.008e+00	9.027e+01
1.04	9.342e+00	6.536e+01	7.299e−02	6.174e+01	4.448e+03	5.542e+02	1.487e+00	1.165e+02
1.05	5.537e−01	3.190e+01	4.178e−04	2.201e−02	5.900e+02	3.495e+02	8.092e−04	4.421e+01
2.06	3.202e+01	6.464e+01	1.873e+00	3.152e+01	9.308e+05	1.415e+03	2.211e+00	1.177e+02
2.07	3.453e+01	1.654e+01	1.371e+01	6.260e+00	1.376e+02	1.234e+02	6.765e+01	4.261e+01
2.08	7.405e+01	6.866e+01	4.320e+00	3.332e+01	6.045e+05	7.049e+03	8.010e+01	1.602e+02
2.09	2.030e+01	5.129e+01	8.313e+00	3.241e+01	3.287e+05	2.260e+02	3.715e+01	1.424e+02
3.10	4.927e+04	1.210e+04	2.472e+03	4.680e+03	1.048e+07	2.325e+05	2.234e+04	6.582e+04
3.11	2.337e+02	8.049e+01	2.610e+02	6.898e+01	5.563e+02	1.885e+02	4.841e+02	1.792e+02
3.12	1.716e+04	1.010e+05	6.880e+00	1.499e+03	6.739e+08	2.964e+07	7.067e+00	1.240e+05
3.13	3.187e+01	1.404e+02	1.805e+00	1.050e+02	3.588e+03	9.766e+02	2.019e+00	1.607e+02
3.14	2.114e−02	2.780e−01	6.335e−04	8.335e−02	1.016e+02	6.893e+00	2.241e−03	3.609e−01
4.15	1.349e+03	1.125e+02	9.561e+01	6.304e+01	3.134e+03	4.190e+02	7.482e+02	2.579e+02
4.16	6.812e+00	5.866e+00	3.164e+00	3.113e+00	2.283e+01	1.475e+01	9.467e+00	8.788e+00
4.17	1.885e+01	1.854e+00	2.314e+01	4.799e−1	2.714e+01	5.368e+00	2.041e+01	1.716e+00
4.18	7.823e+01	5.327e+00	8.362e+01	1.387e+00	1.084e+02	2.034e+01	1.062e+02	8.653e+00
4.19	6.546e+00	2.504e−01	4.953e+00	2.307e−01	2.069e+01	2.504e−01	8.861e+00	2.504e−01
5.20	1.108e+00	1.973e+00	6.419e−01	1.837e+00	1.273e+05	2.104e+02	9.083e−01	2.075e+00
5.21	7.598e+00	5.096e−01	1.803e+00	1.227e−01	8.327e+01	7.596e+00	4.714e+00	1.086e+00
5.22	1.114e+01	2.024e+00	6.919e−01	7.305e−1	8.487e+01	1.260e+01	8.905e+00	6.475e+00
5.23	1.038e+00	1.229e+00	6.147e−01	1.175e+00	2.385e+00	2.263e+00	1.078e+00	1.798e+00
5.24	5.915e+02	1.532e+02	9.995e+01	1.532e+02	1.333e+03	3.299e+02	8.494e+02	3.055e+02

The results of a study of the GDS algorithm for functions of high dimension are presented in Figure 6. The GDS algorithm is executed on the functions 1.02, 1.02, 1.04 and 1.05 of dimensions 100 and 200 to search the global minimum. The error to the global minimum in the logarithmic scale and the CPU time are shown for 10^4, 10^5, 10^6 and 10^7. It is interesting to remark that the CPU time grows linearly with the dimension of the problem.

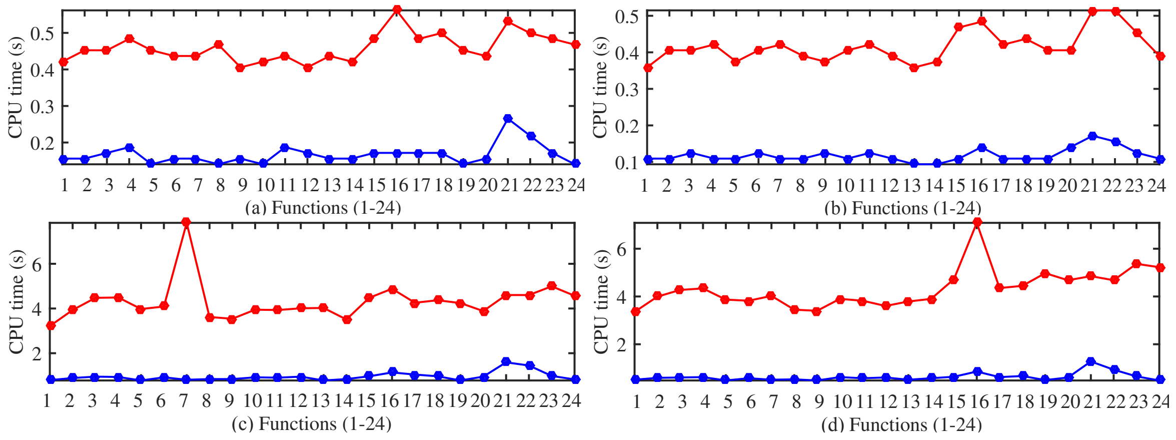

Figure 4. Comparative study of the CPU time of GDS (blue) and DIRECT (red) for solving 24 problems of dimension five (**a**) and dimension 10 (**b**) using 10^3 function evaluations (**c**) and 10^4 function evaluations (**d**).

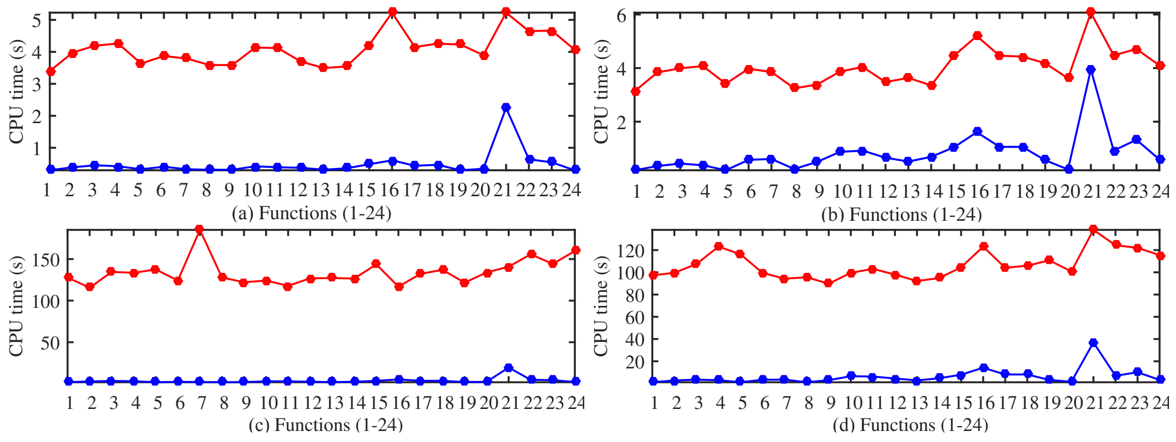

Figure 5. Comparative study of CPU time of GDS (blue) and DIRECT (red) for solving 24 problems of dimension 20 (**a**) and dimension 40 (**b**) using 10^4 function evaluations (**c**) and 10^5 function evaluations (**d**).

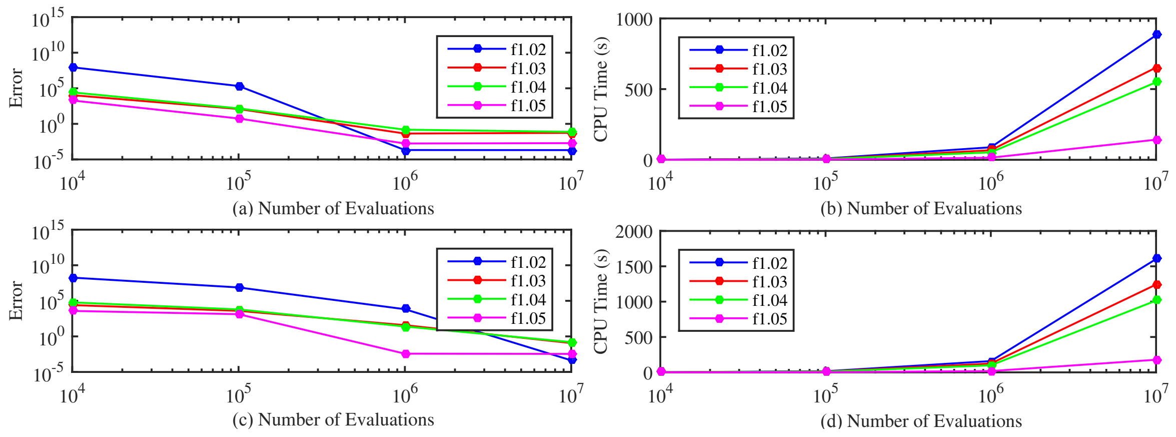

Figure 6. Study of the error to attain the global optimum (**a**) and CPU time (**b**) for functions of dimension 100 (**c**) and dimension 200 (**d**) using the GDS algorithm.

Since GDS is simple and computationally efficient, it can be successfully applied to problems of high dimensions where the number of function evaluations to obtain a good result is large for any direct search algorithm.

5. Conclusions

A direct search algorithm for the global optimization of multivariate functions has been reported. The algorithm performs a sequence of operations for an initial isosceles right n-simplex. The result of these operations is always a similar n-simplex of a different size. The operations depend on the values of the vertices of the n-simplex, where these values are computed using a transformed cost function that improves the exploratory features of the algorithm. The convergence properties of the algorithm have been proven. The results of an extensive experimental study demonstrate that the algorithm is capable of approaching the global optimum for black box problems with moderate computation time. It is competitive with most of the heuristic-based global optimization strategies presently used. Furthermore, it has been compared to DIRECT, a well-known Lipschitzian-based direct search algorithm. The new algorithm is very simple and computationally cost-effective and can be applied to problems of high dimension, because the empirical results demonstrate that its computation time increases linearly with the dimension of the problem. In future work, we plan to extend the algorithm to constrained optimization.

Acknowledgments: The authors thank Instituto de las Tecnologías Avanzadas de la Producción (ITAP) for supporting the publication costs.

Author Contributions: Enrique Baeyens and Alberto Herreros design the GDS algorithm and develop the experiments. José R. Perán analyzed the results. The three autors contributed equally to writing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lewis, R.M.; Torczon, V.; Trosset, M.W. Direct Search Methods: Then and Now. *J. Comput. Appl. Math.* **2000**, *124*, 191–207.
2. Cohn, A.; Scheinberg, K.; Vicente, L. *Introduction to Derivative-Free Optimization*; MOS/SIAM Series on Optimization, SIAM: Philadelphia, PA, USA, 2009.
3. Pinter, J.D. Global Optimization: Software, Test Problems, and Applications. In *Handbook of Global Optimization*; Pardalos, P.M., Romeijn, H.E., Eds.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2002; Volume 2.
4. Kolda, T.G.; Lewis, R.M.; Torczon, V. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Rev.* **2003**, *45*, 385–482.
5. Neumaier, A. Complete Search in Continuous Global Optimization and Constraint Satisfaction. *Acta Numer.* **2004**, *13*, 271–369.
6. Conn, A.R.; Scheinberg, K.; Vicente, L.N. *Introduction to Derivative-Free Optimization*; SIAM: Philadelphia, PA, USA, 2009; Volume 8.
7. Rios, L.M.; Sahinidis, N.V. Derivative-free optimization: A review of algorithms and comparison of software implementations. *J. Glob. Optim.* **2013**, *56*, 1247–1293.
8. Swann, W.H. Direct Search Methods. In *Numerical Methods for Unconstrained Optimization*; Murray, W., Ed.; Academic Press: Salt Lake City, UT, USA, 1972; pp. 13–28.
9. Evtushenko, Y.G. Numerical methods for finding global extrema (case of a non-uniform mesh). *USSR Comput. Math. Math. Phys.* **1971**, *11*, 38–54.
10. Strongin, R. A simple algorithm for searching for the global extremum of a function of several variables and its utilization in the problem of approximating functions. *Radiophys. Quantum Electron.* **1972**, *15*, 823–829.
11. Sergeyev, Y.D.; Strongin, R. A global minimization algorithm with parallel iterations. *USSR Comput. Math. Math. Phys.* **1989**, *29*, 7–15.
12. Dennis, J.E.J.; Torczon, V. Direct Search Methods On Parallel Machines. *SIAM J. Optim.* **1991**, *1*, 448–474.
13. Hooke, R.; Jeeves, T. Direct search solution of numerical and statistical problems. *J. ACM* **1961**, *8*, 12–229.

14. Nelder, J.A.; Mead, R. A Simplex Method for Function Minimization. *Comput. J.* **1965**, *7*, 308–313.
15. Spendley, W.; Hext, G.R.; Himsworth, F.R. Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation. *Technometrics* **1962**, *4*, 441–461.
16. Lagarias, J.; Reeds, J.; Wright, M.; Wright, P. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM J. Optim.* **1998**, *9*, 112–147.
17. Conn, A.; Gould, N.; Toint, P. *Trust-Region Methods*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2000.
18. Regis, R.; Shoemaker, C. Constrained global optimization of expensive black box functions using radial basis functions. *J. Glob. Optim.* **2005**, *31*, 153–171.
19. Custodio, A.; Rocha, H.; Vicente, L. Incorporating minimum Frobenius norm models in direct search. *Comput. Optim. Appl.* **2010**, *46*, 265–278.
20. Conn, A.; Le Digabel, S. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optim. Methods Softw.* **2013**, *28*, 139–158.
21. Lewis, R.; Torczon, V. Pattern search algorithms for bound constrained minimization. *SIAM J. Optim.* **1999**, *9*, 1082–1099.
22. Lewis, R.; Torczon, V. Pattern search algorithms for linearly constrained minimization. *SIAM J. Optim.* **2000**, *10*, 917–941.
23. Lewis, R.; Torczon, V. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM J. Optim.* **2002**, *12*, 1071–1089.
24. Kolda, T.; Lewis, R.; Torczon, V. *A Generating Set Direct Search Augmented Lagrangian Algorithm for Optimization with a Combination of General and Linear Constraints*; Sandia National Laboratories: Livermore, CA, USA, 2006.
25. Audet, C.; Dennis, J.E. A pattern search filter method for nonlinear programming without derivatives. *SIAM J. Optim.* **2004**, *14*, 980–1010.
26. Abramson, M.; Audet, C.; Dennis, J. Filter pattern search algorithms for mixed variable constrained optimization problems. *Pac. J. Optim.* **2007**, *3*, 477–500.
27. Dennis, J.; Price, C.; Coope, I. Direct search methods for nonlinearly constrained optimization using filters and frames. *Optim. Eng.* **2004**, *5*, 123–144.
28. Solis, F.J.; Wets, R.J.B. Minimization by Random Search Techniques. *Math. Oper. Res.* **1981**, *6*, 19–30.
29. Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1984.
30. Michalewicz, Z.; Fogel, D.B. *How to Solve It: Modern Heuristics*, 2nd ed.; Springer: Berlin, Germany, 2004.
31. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680.
32. Bäck, T. *Evolutionary Algorithms in Theory and Pratice*; Oxford University Press: New York, NY, USA, 1996.
33. Larrañaga, P.; Lozano, J.A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*; Kluwer: Boston, MA, USA, 2002.
34. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
35. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–57.
36. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359.
37. Chakraborty, U.K. *Advances in Differential Evolution*; Springer Publishing Company: Berlin, Germany, 2008.
38. Glover, F. Tabu Search—Part I. *ORSA J. Comput.* **1989**, *2*, 190–206.
39. Glover, F.; Laguna, M. *Tabu Search*; Kluwer Academic Publishers: New York, NY, USA, 1997.
40. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41.
41. Dorigo, M.; Stützle, T. *Ant Colony Optimization*; Bradford Company: Scituate, MA, USA, 2004.
42. Paulavičius, R.; Žilinskas, J. *Simplicial Global Optimization*; Springer: Berlin, Germany, 2014.
43. Sergeyev, Y.D.; Strongin, R.G.; Lera, D. *Introduction to Global Optimization Exploiting Space-Filling Curves*; Springer: Berlin, Germany, 2013.
44. Strongin, R.G.; Sergeyev, Y.D. *Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms*; Springer: Berlin, Germany, 2013.
45. Zhigljavsky, A.; Žilinskas, A. *Stochastic Global Optimization*; Springer: Berlin, Germany, 2008.
46. Zhigljavsky, A.A. *Theory of Global Random Search*; Springer: Berlin, Germany, 2012.

47. Shubert, B.O. A sequential method seeking the global maximum of a function. *SIAM J. Numer. Anal.* **1972**, *9*, 379–388.
48. Sergeyev, Y.D.; Kvasov, D.E. Global search based on efficient diagonal partitions and a set of Lipschitz constants. *SIAM J. Optim.* **2006**, *16*, 910–937.
49. Lera, D.; Sergeyev, Y.D. Lipschitz and Hölder global optimization using space-filling curves. *Appl. Numer. Math.* **2010**, *60*, 115–129.
50. Kvasov, D.E.; Sergeyev, Y.D. Lipschitz global optimization methods in control problems. *Autom. Remote Control* **2013**, *74*, 1435–1448.
51. Jones, D.R.; Perttunen, C.D.; Stuckman, B.E. Lipschitzian optimization without the Lipschitz constant. *J. Optim. Theory Appl.* **1993**, *79*, 157–181.
52. Gablonsky, J.M.; Kelley, C.T. A locally-biased form of the DIRECT algorithm. *J. Glob. Optim.* **2001**, *21*, 27–37.
53. Finkel, D.E.; Kelley, C. Additive scaling and the DIRECT algorithm. *J. Glob. Optim.* **2006**, *36*, 597–608.
54. Paulavičius, R.; Sergeyev, Y.D.; Kvasov, D.E.; Žilinskas, J. Globally-biased Disimpl algorithm for expensive global optimization. *J. Glob. Optim.* **2014**, *59*, 545–567.
55. Liu, Q.; Cheng, W. A modified DIRECT algorithm with bilevel partition. *J. Glob. Optim.* **2014**, *60*, 483–499.
56. Ge, R.P. The theory of filled function-method for finding global minimizers of nonlinearly constrained minimization problems. *J. Comput. Math.* **1987**, *5*, 1–9.
57. Ge, R.; Qin, Y. The globally convexized filled functions for global optimization. *Appl. Math. Comput.* **1990**, *35*, 131–158.
58. Han, Q.; Han, J. Revised filled function methods for global optimization. *Appl. Math. Comput.* **2001**, *119*, 217–228.
59. Liu, X.; Xu, W. A new filled function applied to global optimization. *Comput. Oper. Res.* **2004**, *31*, 61–80.
60. Zhang, Y.; Zhang, L.; Xu, Y. New filled functions for nonsmooth global optimization. *Appl. Math. Model.* **2009**, *33*, 3114–3129.
61. Wolpert, D.; Macready, W. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82.
62. Hansen, N.; Finck, S.; Ros, R.; Auger, A. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions*; Research Report RR-6829; INRIA: Rocquencourt, France, 2009.
63. Mathworks. *MATLAB User's Guide (R2010a)*; The MathWorks, Inc. Natick, MA, USA, 2010.
64. Finkel, D.E. *DIRECT Optimization Algorithm User Guide*; Center for Research in Scientific Computation, North Carolina State University: Raleigh, NC, USA, 2003.