A simplicial homology algorithm for Lipschitz optimisation

Stefan Endres

A simplicial homology algorithm for Lipschitz optimisation

by

Stefan Endres

A dissertation submitted in partial fulfilment
of the requirements for the degree

Master of Engineering (Chemical Engineering)

in the

Department of Chemical Engineering
Faculty of Engineering, the Built Environment and Information Technology

University of Pretoria
Pretoria

October 2017

Synopsis

The simplicial homology global optimisation (SHGO) algorithm is a general purpose global optimisation algorithm based on applications of simplicial integral homology and combinatorial topology. SHGO approximates the homology groups of a complex built on a hypersurface homeomorphic to a complex on the objective function. This provides both approximations of locally convex subdomains in the search space through Sperner's lemma (Sperner, 1928) and a useful visual tool for characterising and efficiently solving higher dimensional black and grey box optimisation problems. This complex is built up using sampling points within the feasible search space as vertices. The algorithm is specialised in finding all the local minima of an objective function with expensive function evaluations efficiently which is especially suitable to applications such as energy landscape exploration. SHGO was initially developed as an improvement on the topographical global optimisation (TGO) method first proposed by Törn (1986; 1990; 1992). It is proven that the SHGO algorithm will always outperform TGO on function evaluations if the objective function is Lipschitz smooth. In this paper SHGO is applied to non-convex problems with linear and box constraints with bounds placed on the variables. Numerical experiments on linearly constrained test problems show that SHGO gives competitive results compared to TGO and the recently developed Lc-DISIMPL algorithm (Paulavičius and Žilinskas, 2016) as well as the PSwarm and DIRECT-L1 algorithms. Furthermore SHGO is compared with the TGO, basinhopping (BH) and differential evolution (DE) global optimisation algorithms over a large selection of black-box problems with bounds placed on the variables from the SciPy (Jones, Oliphant, Peterson, et al., 2001–) benchmarking test suite. A Python implementation of the SHGO and TGO algorithms published under a MIT license can be found from `https://bitbucket.org/upiamcompthermo/shgo/`.

Keywords: Global optimisation, SHGO, Computational homology

Mathematics Subject Classification (2010) 90C26 Nonconvex programming, global optimisation

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Objective function statement and nomenclature

Consider a general optimisation problem of the form

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{s.t. } \mathbf{g}(\mathbf{x}) \geq 0 \end{aligned} \tag{1.1}$$

The continuous real objective function $f(\mathbf{x})$ of n dimensionality can be either smooth or non-smooth depending on the local minimisation method used. The variables $\mathbf{x}$ are assumed to be bounded. In this publication we mainly consider real, smooth, but not necessarily convex functions with linear constraint functions. In addition we will assume that the objective function has a finite number of local minima

$$f : \mathbb{R}^n \to \mathbb{R} \tag{1.2}$$

$\mathbf{g}$ maps the set of linear constraints

$$\mathbf{g} : [\mathbf{l}, \mathbf{u}]^n \to \mathbb{R}^m \tag{1.3}$$

for example if lower and upper bounds l_i and u_i are implemented for each variable then we have an initially defined hyperrectangle

$$\mathbf{x} \in \Omega \subseteq [\mathbf{l}, \mathbf{u}]^n \ = \ [l_1,\ u_1] \ \times \ [l_2,\ u_2] \ \times \ \ldots \ \times \ [l_n,\ u_n] \subseteq \mathbb{R}^n \tag{1.4}$$

where Ω is the limited feasible subset excluding points outside the bounds and constraints.

$$\Omega = \{\mathbf{x} \in [\mathbf{l}, \mathbf{u}]^n \mid \mathbf{g}_i(\mathbf{x}) \geq 0, \forall i = 1, \ldots, m\} \tag{1.5}$$

Since the constraints in $\mathbf{g}$ are linear the set Ω is always a compact space.

In the development of SHGO several concepts from algebraic and combinatorial topology Henle (1979) are required. The following definition was adapted from Hatcher (2002: p. 9)

Definition 1. *A* **k-*simplex*** *is a set of $n+1$ vertices in a convex polyhedron of dimension n. Formally if the $n+1$ points are the $n+1$ standard $n+1$ basis vectors for $\mathbb{R}^{(n+1)}$. Then the n-dimensional k-simplex is the set*

$$S^n = \left\{ (t_1, \ldots, t_{n+1}) \in \mathbb{R}^{n+1} \mid \sum_{1}^{n+1} t_{n+1} = 1, t_i \geq 0 \right\}$$

For example, a 2-simplex is a triangle and a 3-simplex is a tetrahedron. We will use the following combinatorial definition of a simplicial complex (Hatcher, 2002: p. 107)

Definition 2. *A* ***simplicial complex*** *$\mathcal{H}$ is a set $\mathcal{H}^0$ of vertices together with sets $\mathcal{H}^n$ of n-simplices, which are $(n+1)$-element subsets of $\mathcal{H}^0$. The only requirement is that each $(k+1)$-elements subset of the vertices of an n-simplex in $\mathcal{H}^n$ is a k-simplex, in $\mathcal{H}^k$.*

Thus each n-simplex has $n+1$ distinct vertices, and no other n-simplex has this same set of vertices.

In this publication the $\mathcal{H}$ symbol will be used to represent a (finite) simplicial complex rather than the more standard Δ to avoid confusion with the difference and Laplacian operators common in optimisation. The superscript $\mathcal{H}^k$ represents the subset of $k-$dimensional simplices where for an n dimensional problem the highest dimensional $k-$simplex contains $n+1$ vertices. Finally we define a k-chainHenle (1979)

Definition 3. *A* **k-*chain*** *is a union of simplices.*

For example a 0-chain is a set of vertices, a 1-chain is a set of edges and a 2-chain is a set of triangles. $C(\mathcal{H}^k)$ denotes a $k-$chain of $k-$simplices. A vertex in $\mathcal{H}^0$ is denoted by v_i. If v_i and v_j are two endpoints of a directed edge in $\mathcal{H}^1$ from v_i to v_j then the symbol $\overline{v_i v_j}$ represents the edge so that it is bounded by the $0-$chain $\partial\left(\overline{v_i v_j}\right) = v_j - v_i$ and similarly for an edge directed from v_j to v_i, we have, $\partial\left(\overline{v_j v_i}\right) = \partial\left(-\overline{v_i v_j}\right) = v_i - v_j$. Higher dimensional simplices can be represented and directed in a similar manner, for example a triangle consisting of three vertices v_i, v_j and v_k directed as $\overline{v_i v_j v_k}$ has the boundary of directed edges $\partial\left(\overline{v_i v_j v_j}\right) = \overline{v_i v_j} + \overline{v_j v_k} + \overline{v_j v_i}$.

1.2 Multimodal objective functions and local minima mapping

Non-convex problems are commonly solved using global optimisation methods. One such example is the topographical global optimisation (TGO) method Henderson, de Sá Rêgo, Sacco, and Rodrigues (2015); Törn (1986); Törn (1990); Törn & Viitanen (1992) which is a clustering algorithm that finds several local minima from which the (approximate) global minimum is found. It is often desirable to find all the local minima of the objective function for example in applications such as energy landscape exploration of potential models wherein mapping the local minima of the potential functions can provide valuable insights into the system. Algorithms such as the basin-hopping global optimisation algorithm are typically used to find these points (Wales, 2015).

The graph extracted from the topographical global optimisation (TGO) Henderson et al. (2015); Törn (1986); Törn (1990); Törn & Viitanen (1992) topograph (as described in chapter 2) is unsatisfactory in some ways. Primarily because several starting points in the same locally convex domain can be generated even when enough information from the objective function sampling is known to prevent this from occurring. This leads to superfluous function evaluations in the local minimisation step of the algorithm. Contrary to intuition, this problem is exacerbated by increasing the number of initial sampling points used in the algorithm as demonstrated in chapter 2. This can lead to a very large number of function evaluations required to solve the problem. In particular in multimodal energy surfaces where the local minima can often be located in short distances relative to the search space Zhang and Rangaiah (2011) and thus requires a large number of initial sampling to locate all these domains. Some shortcomings in using the TGO method to map local minima are:

- Geometric information available from the sampling points is being disregarded by the graphs built up using only the Euclidean distance metric.
- Knowledge of the number and location of local minimisers in a given sampling set is not being used to the full extent.
- More than one minimiser might be produced in the same locally convex domain and there is no guarantee that a minimiser set produced by TGO will be in the locally convex domains of all local minima even if the number of local minima is known and a minimiser set of this cardinality is produced.

By constructing a directed simplicial complex we show that the simplicial homology global optimisation (SHGO) algorithm does not produce superfluous starting points for the class of all Lipschitz smooth functions resulting in more efficient performance for these problems compared to TGO. The directed complex is also used to approximate the homology group of the objective function hypersurface which, using integral homology version of the Invariance Theorem Henle (1979), allows for efficient mapping of optimisation problems where the number of local minima is known *a-priori*.

1.3 Derivative-free methods for Lipschitz optimsation problems

Both the SHGO and TGO algorithms only make use of function evaluations without requiring the derivatives of objective functions. This makes them applicable to black-box global optimisation problems. A recent review and experimental comparison of 22 derivative-free optimisation algorithms by Rios and Sahinidis (2013) concluded that global optimisation solvers solvers such as TOMLAB/MULTI-MIN, TOMLAB/GLCCLUSTER, MCS and TOMLAB/LGO perform better, on average, than other derivative-free solvers in terms of solution quality within 2500 function evaluations. Both the TOMLAB/GLC-CLUSTER and MCS Huyer and Neumaier (1999) implementations are based on the well-known DIRECT (DIviding RECTangle) algorithm Jones, Perttunen, and Stuckman (1993).

The DISIMPL (DIviding SIMPLices) algorithm was recently proposed by Paulavičius and Žilinskas (2014b). The experimental investigation in Paulavičius & Žilinskas (2014b) shows that the proposed simplicial algorithm gives very competitive results compared to the DIRECT algorithm. DISIMPL has been extended in Paulavičius and Žilinskas (2014a); Paulavičius, Sergeyev, Kvasov, and Žilinskas (2014). The Gb-DISIMPL (Globally-biased DISIMPL) was compared in Paulavičius et al. (2014) to the DIRECT and DIRECTl methods in extensive numerical experiments on 800 multidimensional multiextremal test functions.

In a recent adaption of DISIMPL for linearly constrained optimisation problems, Lc-DISIMPL Paulavičius & Žilinskas (2016) showed extremely competitive results compared to the PSwarm Vaz and Vicente (2009) and DIRECT-L1 algorithms Finkel (2003). In particular the Lc-DISIMPL-v algorithm was shown to solve the problems in a fewer number of function evaluations on average and was the only algorithm to converge on all of the test problems. In this publication both the SHGO and TGO algorithms were tested on the same problem set and the results are compared to the data from Paulavičius & Žilinskas (2016) which also contains results on the PSwarm Vaz & Vicente (2009) and DIRECT-L1 algorithms Finkel (2003).

The DISIMPL algorithm is the most similar to SHGO in the sense that both make use of a simplicial complex. DISIMPL uses a simplicial complex in a spatial partitioning of the initial search space. Since the geometric structure of the two algorithms are related, it is reasonable to expect some theoretical relation of its properties. In particular the graph structure in the DISIMPL-v algorithm Paulavičius & Žilinskas (2016) can be used to build the directed simplicial complex used by SHGO. In chapter 4 we also show how some of the same principles developed for SHGO can also be applied in the DISIMPL-v algorithm since the same information is readily available to the algorithm.

1.4 Overview of this publication

The TGO method is briefly reviewed in chapter 2 closely following the formalism developed by Henderson et al. (2015). In chapter 3 we provide numerical examples of TGO which is then used as an informal experimental motivation for extending the algorithm. These two sections are important for continuity and understanding of the improved features of SHGO, in particular Definition 9 which will be used as a performance criterion. In section 3.1 we present the most immediately apparent extension of TGO and illustrate the shortcomings of that approach. The new SHGO method is then formally presented in chapter 4. In chapter 5 we provide experimental results of linearly constrained problems comparing the SHGO, TGO, Lc-DISIMPL Paulavičius & Žilinskas (2016), PSwarm Vaz & Vicente (2009) and DIRECT-L1 Finkel (2003) algorithms. Furthermore SHGO is compared with the TGO, basinhopping (BH) and differential evolution (DE) global optimisation algorithms over a large selection of black-box problems from the SciPy (Jones, Oliphant, Peterson, et al., 2001–) global optimisation benchmarking test suite. We conclude with various recommendations for possible further improvements of SHGO.

CHAPTER 2

Topographical Global Optimisation (TGO)

The Topographical Global Optimisation (TGO) was originally conceived by Törn (1990) and Henderson et al. Henderson et al. (2015); Henderson, de Sá Rêgo, and Imbiriba (2017) introduced new formalisms and empirical methods to determine hyperparameters described in this section. Henderson et al. (2015) also presents the algorithm in an introductory fashion. It is in essence an iterative clustering algorithm that maps the hypersurface of the objective function into a topography matrix (called a t-matrix) and then finds a certain number of starting points referred to as local minimisers. A local search using the local minimisers as starting points is then used to find each minimum from which the global minimum is finally calculated. Henderson et al. (2015) used the feasible direction interior-point method proposed by Herskovits (1998) in this step. The feasible direction interior-point method allows for minimisation of problems with linear and/or nonlinear equality constraints; an extension by Henderson et al. (2015) of the original applications of Törn (1990). The TGO method consists of three steps:

1. Uniform random sampling generation of N points in the search space.
2. Construction of the topograph, which is a directed graph with the sampled points as vertices on a k-nearest neighbours basis with the direction of the arc directed towards a point with a larger function value.
3. Local minimisation of topograph minimisers.

2.1 Step 1: Random Sampling Point generation

In order to generate the uniform sampling points within Ω the deterministic Sobol sequence is used in this publication Henderson et al. (2015); Sobol (1967). Other possible low discrepancy sequences such as the Halton and Van der Corput sequences Kuipers and

Niederreiter (1974) can also be used in this step. An efficient Gray code implementation was proposed by Antonov and Saleev (1979) wherein a single XOR operation for each dimension can be used to find the next sampling point in the sequence $x_{n,i} = x_{n-1,i} \oplus v_{k,i}$. An adaptation of this method is available in the open source Python library UQToolboxBigoni (2016). The Sobol sequenced points are generated within the n dimensional hypercube $[0,1]^n \in \mathbb{R}^n$, providing a uniform distribution on the hypersurface within this space. In the current implementation this set of points is stretched across the lower and upper bounds to form the hyperrectangle $[\mathbf{l}, \mathbf{u}]^n = [l_1,\ u_1] \times\ [l_2,\ u_2] \times \cdots \times [l_n,\ u_n] \subseteq \mathbb{R}^n$. The subset of feasible points contained in Ω is found by discarding any points lying outside the constraints $\mathbf{g}(\mathbf{x}) > 0$.

2.2 Step 2: Construction of the topograph

The topograph is constructed from the generated sampling points within Ω. From the topograph several global minimisers in f are found using the definitions developed in this section which are then used as starting points for local minimisation routines. First N points are selected from the uniformly generated sequence of points within the feasible domain of $\Omega \subset \mathbb{R}^n$. Points generated by the sequence that lie outside the constraints are excluded. The points are denoted by $\mathbf{p}_i, i = 1, 2, 3 \ldots N$. Next for each point $\mathbf{p}_i$ a reference list is constructed by ordering the other $N-1$ points from their nearest to farthest Euclidean distances. These ordered lists make up the rows of the topography matrix (or topograph). Furthermore, for some point $\mathbf{p}_j \in \{1, 2, 3 \ldots (N-1)\}$ in the row with the first entry $\mathbf{p}_i$, a sign is assigned as follows:

$$\text{sign}(\mathbf{p}_j) = \begin{cases} f(\mathbf{p}_j) \geq f(\mathbf{p}_i) & \rightarrow + \\ f(\mathbf{p}_j) < f(\mathbf{p}_i) & \rightarrow - \end{cases}$$

In order to demonstrate this construction we will define this ordered list in such a way that the increasing indices represent an ordered list of the nearest points to $\mathbf{p}_1$, that is $||\mathbf{p_i} - \mathbf{p_{i+1}}|| \leq ||\mathbf{p_{i+1}} - \mathbf{p_{i+2}}|| \, \forall i$. Suppose for example that $f(\mathbf{p}_2) \geq f(\mathbf{p}_1)$, $f(\mathbf{p}_3) < f(\mathbf{p}_1)$ and $f(\mathbf{p}_N) \geq f(\mathbf{p}_1)$, the resulting topograph with the first row known is:

$$t\text{-matrix} = \left(\begin{array}{c|cccc} \mathbf{p}_1 & +\mathbf{p}_2 & -\mathbf{p}_3 & \cdots & +\mathbf{p}_N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_N & \mathbf{p}_j & \cdots & \mathbf{p}_j & \mathbf{p}_j \end{array} \right) \tag{2.1}$$

Note that the remaining rows (represented by unknown points and signs $\mathbf{p}_j$) are constructed similarly to the first row for every $\mathbf{p}_i$ row. The topography matrix can be interpreted as a directed graph, where the signs represent the directed arcs on the graph. It should also be noted that if $\mathbf{g}$ contains non-linear constraints then the graphs produced by the topograph may be connected across disconnected and/or non-convex subspaces of Ω. Example 1 in chapter 3 demonstrates the construction of the topograph numerically.

Given and integer $1 \leq k \leq (N-1)$, the $N \times k$ submatrix obtained by considering only the k-nearest neighbours is called the k-t-matrix. For example for $k = 1$:

$$1\text{-}t\text{-matrix} = \left(\begin{array}{c|c} \mathbf{p}_1 & +\mathbf{p}_2 \\ \vdots & \vdots \\ \mathbf{p}_N & \mathbf{p}_j \end{array}\right) \tag{2.2}$$

for $k = 2$:

$$2\text{-}t\text{-matrix} = \left(\begin{array}{c|cc} \mathbf{p}_1 & +\mathbf{p}_2 & -\mathbf{p}_3 \\ \vdots & \vdots & \vdots \\ \mathbf{p}_N & \mathbf{p}_j & \mathbf{p}_j \end{array}\right) \tag{2.3}$$

and so forth. The k-t-matrix is a representation of its k^+-topograph where every row forms a directed subgraph.

The following definitions adapted from Henderson et al. (2015) are used to find the global minimisers of the objective function

Definition 4. *Given an integer $1 \leq k \leq (N-1)$, the ith row of the k-t-matrix is said to be a positive row, if all its elements have a plus sign. That is iff $f(\mathbf{p}_j) \geq f(\mathbf{p}_i)\ \forall j$.*

Definition 5. *Given an integer $1 \leq k \leq (N-1)$, a sampling point $\mathbf{p}_i$ has a positive reference in the k-t-matrix, if there exists $j \neq i$ such that (a) the j^{th} row of the k-t-matrix is a positive row and (b) the number $+i$ is an element of this j^{th} row.*

Definition 6. *Given an integer $1 \leq k \leq (N-1)$, the sample point $\mathbf{p}_i$ is called a local minimiser of f in the k^+-topograph if the i^{th} row of the k-t-matrix is a positive row.*

Definition 7. *Given an integer $1 \leq k \leq (N-1)$, the sample point $\mathbf{p}_i$ is a global minimiser of f in the k^+-topograph if $\mathbf{p}_i$ is a local minimiser of f in the k^+-topograph and, in addition, $\mathbf{p}_i$ has no positive references in the k-t-matrix.*

The following propositions can be readily demonstrated to show the consistency of the aforementioned definitions Henderson et al. (2015).

Proposition 1. *Given an integer $1 \leq k \leq (N-1)$, the sample point $\mathbf{p}_i$ is a global minimiser of f in the k^+-topograph if and only if the sample point $\mathbf{p}_i$ is the only minimiser of f in the k^+-topograph which is global.*

Proposition 2. *Given an integer $1 \leq k \leq (N-1)$, then the ith row of k-t-matrix is the only positive row of this matrix if and only if the sample point $\mathbf{p}_i$ is the only minimiser of f in the k^+-topograph which is global.*

Corollary 1. *Given an integer $1 \leq k \leq (N-1)$, if the sample point $\mathbf{p}_i$ is the only local minimiser of f in the k^+-topograph, then $\mathbf{p}_i$ is a global minimiser of f in this graph.*

In this publication we will use the paradigm that all local minimisers of f in the k^+ -topograph will be used for the local search (Paradigm 2.2 in Henderson et al. (2015)). As described in Törn & Viitanen (1992) the number of local minimisers of f in the k^+-topograph is greater than or equal to number of global minimisers in the topograph. We will therefore employ the following definition

Definition 8. *Given an integer $1 \leq k \leq (N-1)$, the minimiser pool $\mathcal{M}^k$ is the set containing all local minimisers $\mathbf{p}_i$ in the in the k^+-topograph. The total number of starting points used in the local search step is equal to the cardinality of the minimiser pool $|\mathcal{M}^k|$.*

The entire point of using k-t-matrices is because a t-matrix will always have at most one local (and thus global) minimiser. This is undesirable since this sampling point is not necessarily the starting point closest to the true global minimum of the objective function. Henderson et al. (2015) developed a semi-empirical formula producing an integer value k_c which is used as an estimate for the optimal value for the integer k.

2.3 Step 3: Local minimisation

Each of the minimisers from the k_c-t-topograph is now used as a starting point in a local minimisation routine. The resulting minima are used to find the global minimum. Conceivably various local optimisation routines can be used to address a broad class of optimisations problems. For problems with non-linear inequality constraints Henderson et al. (2015) used the feasible direction interior-point method proposed by Herskovits (1998) minimising the objective function f subject to the set of inequality constraint functions $\mathbf{g}$ using the minimiser set as the initial starting points for the algorithm. An algorithm used to solve the feasible direction interior-point method using the set of starting points calculated in step 2 is presented in detail by Henderson et al. (2015).

In this publication we will mainly be using the sequential least squares quadratic programming optimisation algorithm (SLSQP) contained in the SciPy library originally developed by Kraft Kraft (1988, 1994). Our Python implementation of the TGO algorithm published under an open source licence uses this algorithm as implemented in the SciPy library Endres (2016–b); Jones et al. (2001–).

CHAPTER 3

Motivation and a one-dimensional prelude

In this section we will demonstrate how the Euclidean distance criterion in the TGO method disregards useful information about the (approximate) geometry of the objective function and we show how known information can be used effectively both in global optimisation and in mapping the local minima of objective functions as efficiently as possible. We also show how two important hyperparameters used by TGO, namely the number of sampling points N and the choice of k can be iteratively selected by intelligently exploiting information known from the topograph. This draws parallels to other works on iterative versions of TGO (I-TGO) Törn and Viitanen (1996) trying to extract information from black-box objective functions. The informal, but intuitive ideas developed here will later be extended more rigorously to higher dimensional surfaces. Note that from Equation (1.5) Ω is always a compact space, this fact is important in several proofs used in this Section.

Example 1 Consider the following objective function

$$\min_x f(x) = \frac{\sin(x)}{x}, \; x \in \Omega = [1, 20] \tag{3.1}$$

In this instance of the bounded optimisation problem there are 3 local minima which we will try to map in as few function evaluations as possible.

Following the TGO procedure we start by generating low-discrepancy sampling points. The first $N = 10$ points in the 1-dimensional Sobol sequence is given by $\mathcal{P} = \{p_1 = 1.0, p_2 = 10.5, p_3 = 15.25, p_4 = 5.75, p_5 = 8.125, p_6 = 17.625, p_7 = 12.875, p_8 = 3.375, p_9 = 4.5625, p_{10} = 14.0625\} \subset \Omega$. After mapping the objective function at the set of sampling points

$$f : \begin{bmatrix} p_1 = 1.0 \\ p_2 = 10.5 \\ p_3 = 15.25 \\ p_4 = 5.75 \\ p_5 = 8.125 \\ p_6 = 17.625 \\ p_7 = 12.875 \\ p_8 = 3.375 \\ p_9 = 4.5625 \\ p_{10} = 14.0625. \end{bmatrix} \rightarrow \begin{bmatrix} f_1 = 0.84147 \\ f_2 = -0.08378 \\ f_3 = 0.02899 \\ f_4 = -0.08840 \\ f_5 = 0.11858 \\ f_6 = -0.05337 \\ f_7 = 0.02359 \\ f_8 = -0.06853 \\ f_9 = -0.21672 \\ f_{10} = 0.07091 \end{bmatrix} \tag{3.2}$$

the corresponding topograph is constructed

$$\left[\begin{array}{c|ccccccccc} p_1 & -p_8 & -p_9 & -p_4 & -p_5 & -p_2 & -p_7 & -p_{10} & -p_3 & -p_6 \\ p_2 & +p_5 & +p_7 & +p_{10} & +p_3 & -p_4 & -p_9 & +p_6 & +p_8 & +p_1 \\ p_3 & +p_{10} & -p_6 & -p_7 & -p_2 & +p_5 & -p_4 & -p_9 & -p_8 & +p_1 \\ p_4 & -p_9 & +p_5 & +p_8 & +p_1 & +p_2 & +p_7 & +p_{10} & +p_3 & +p_6 \\ p_5 & -p_2 & -p_4 & -p_9 & -p_7 & -p_8 & -p_{10} & +p_1 & -p_3 & -p_6 \\ p_6 & +p_3 & +p_{10} & +p_7 & -p_2 & +p_5 & -p_4 & -p_9 & -p_8 & +p_1 \\ p_7 & +p_{10} & -p_2 & +p_3 & +p_5 & -p_6 & -p_4 & -p_9 & -p_8 & +p_1 \\ p_8 & -p_9 & +p_1 & -p_4 & +p_5 & -p_2 & +p_7 & +p_{10} & +p_3 & +p_6 \\ p_9 & +p_4 & +p_8 & +p_1 & +p_5 & +p_2 & +p_7 & +p_{10} & +p_3 & +p_6 \\ p_{10} & -p_3 & -p_7 & -p_2 & -p_6 & +p_5 & -p_4 & -p_9 & -p_8 & +p_1 \end{array}\right] \tag{3.3}$$

The sampling points together with the objective function evaluations are plotted in Figure 3.1. Using the empirical relation from Henderson et al. (2015) the optimal k_c is calculated at $k_c = 8$. Using Definition 6 we find that the resulting 8-t-matrix has only one minimiser; the global minimiser at $p_9 = 4.5625$. For the local minimisation we use the SLSQP method as implemented in the function scipy.optimize.minimize Jones et al. (2001–) to find the approximate global minimum at $x = 4.4934$.

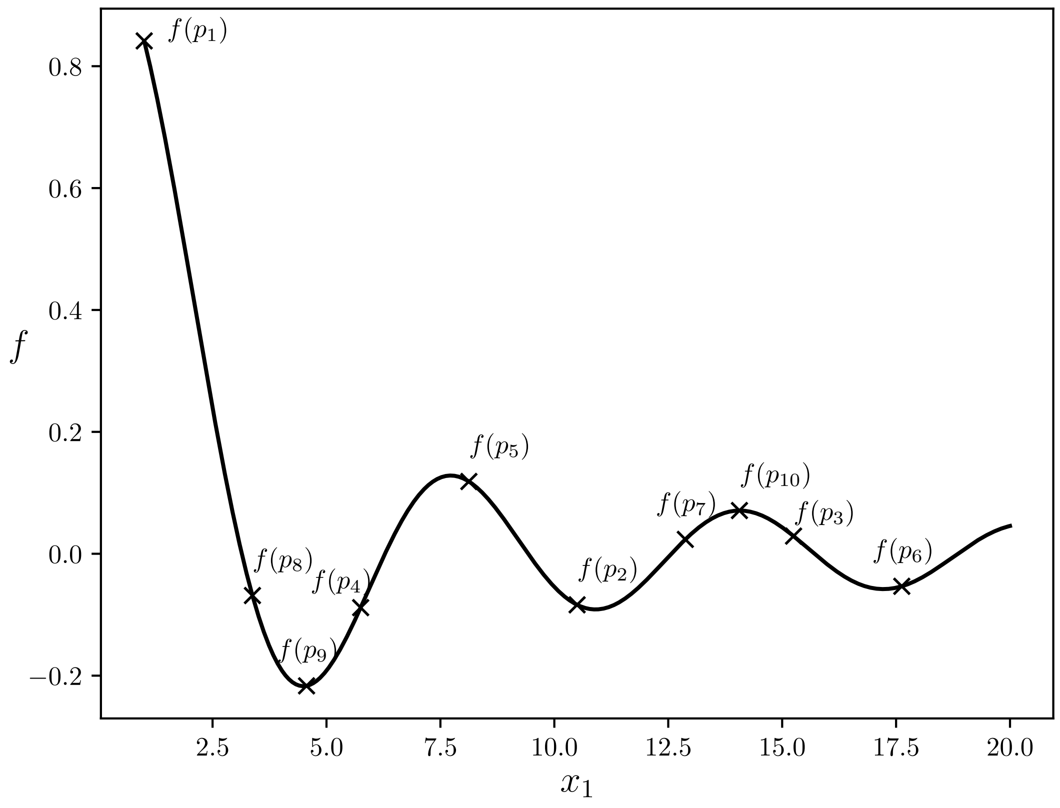

Figure 3.1: Test function give by Equation (3.1) with 10 Sobol sequenced sampling points

Observing Figure 3.1 it is immediately apparent that the set of 10 sampling points alone provides adequate information to deduce that there are at least 3 local minima. Observe that there are at least two other local minima since $f(p_5) < f(p_2) < f(p_7)$. So at least one local minimum exists in the domain $(p_5, p_7) \subset \mathbb{R}$ since between p_5 and p_2 we must have, by the mean value theorem (MVT), $\frac{df}{dx} < 0$ for some domain $x \in [p_5, p_2] \subset \mathbb{R}$. Similarly for $x \in [p_2, p_7] \subset \mathbb{R}$ we have by MVT $\frac{df}{dx} > 0$. Since f is a smooth, continuous function for $x \in (0, \infty)$ there must exist at least one stationary point $x \in (p_5, p_7) \subset \mathbb{R}$ where $\frac{df}{dx} = 0$. Furthermore we observe $f(p_6) < f(p_3)$ indicating another minimum in the domain $x \in (p_3, 20] \subset \mathbb{R}$ since the minimum must be either on the boundary or in $x \in (p_3, 20] \subset \mathbb{R}$ by the same argument as above.

The empirical relation by Henderson et al. (2015) was mainly developed for the purpose of finding the global minimum. Therefore if only 10 sampling points are available, then to find more local minima using the TGO method is required to force a lower k value. Alternatively, since k_c is a function of N, simply sampling more points is sufficient to find all the local minima using Henderson's formula for this test problem. For example at $N = 16$ all 3 local minima are produced by TGO with Henderson's formula. Figure 3.2 shows the number of minimisers found at different k values for this example.

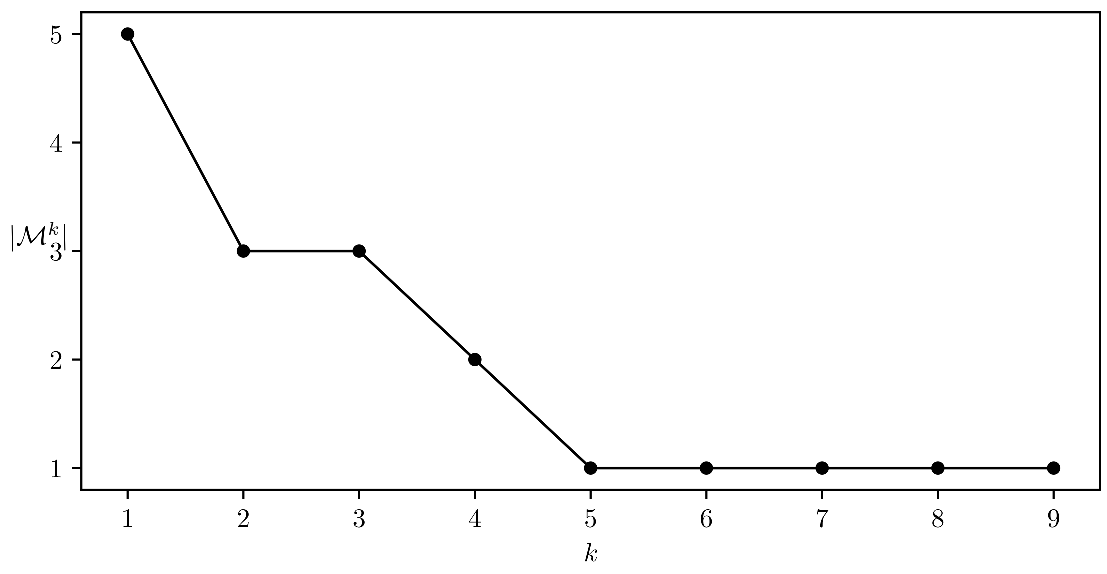

Figure 3.2: Number of minimisers $|\mathcal{M}^k|$ found using the TGO method for different k values at $N = 10$

The maximum minimiser set (other than using every sampling point as a starting point) can be trivially extracted by setting $k = 1$ and calculating $|\mathcal{M}^1|$. However, in this Example it leads to more starting points than optimal since at least two minimisers will be in the same convex basin domain and therefore converge to same minimum in the local minimisation step. This results in superfluous function evaluations without extracting more useful information from the objective function.

This idea drives the motivation behind the following definition.

Definition 9. *For a given set $\mathcal{P}$ of N sampling points, k_{opt} is any integer $1 \leq k \leq (N-1)$ that will produce the optimal minimiser set $\mathcal{M}^{k_{opt}}$ containing the maximum set of minimisers such that no two starting points extracted from $\mathcal{M}^{k_{opt}}$ will lead to the same minimum in the local optimisation step for some tolerance ϵ. In other words every element contained in $\mathcal{M}^{k_{opt}}$ should lie in a unique locally convex sub-domain.*

Note that for a given N, $\mathcal{M}^{k_{opt}}$ might not produce all the true local minima of an objective function. What's important is that, given the information known from the sampling, the maximum number of local minima are found. In addition, no function evaluations are wasted in the local minimisation step which lead to the same minimum.

In Example 1 for $N = 10$ the optimal k values are $k_{opt} = \{2, 3\}$ which will produce 3 minimisers $|\mathcal{M}^2| = |\mathcal{M}^3| = 3$. We will now show that these lower k values carry unexploited information on the best approximate geometry of the objective function. For example in Figure 3.3 we plot the $|\mathcal{M}^k|$ values corresponding to the set $k = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ for every sampling point range $N \in [2, 50]$.

From Figure 3.3 we notice the special property of $k = 3$ for one dimensional objective

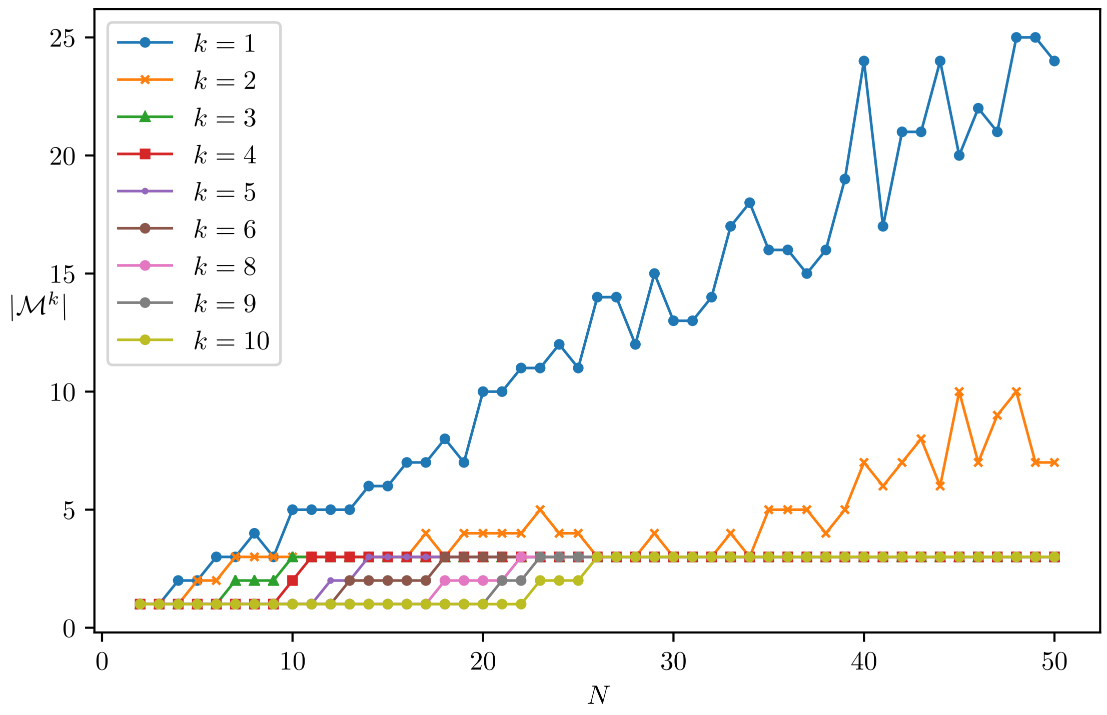

Figure 3.3: Number of minimisers $|\mathcal{M}^k|$ found using the TGO method for the given k values at various sampling points N

functions sampled with the Sobol sequence.

Firstly, for a lower number of sampling points N it provides a higher number of starting minimisers than $k > 3$. Note that by inspection of Definition 6 it can be determined that any $k > 3$ value will always produce an equal or lower number of minimisers than $k = 3$ (this is also true for any $k > i$). When adding columns to a positive row there are only two possibilities: the next sampling point in the row can either have a positive or a negative sign. All other elements in the row have a positive sign by definition (see Definition 6). If the next sampling point in the row has a positive sign then the row will just remain a positive row and the number of minimisers remain the same. If the point is a negative reference point then the row will no longer be a positive row and thus the point is no longer a minimiser, lowering the total.

Secondly it can be observed that $k = 3$ never calculates a number of starting minimisers higher than optimal unlike $k < 3$. Therefore by using $k = 3$ in Example 1 TGO will always find as many minimisers in as few sampling[1] function evaluations as possible and furthermore all local minima will be found when $N \geq 10$. It should be noted that the total number of function evaluations depends on the particular local minimisation algorithm used. However, it is apparent that each minimiser starting point is in a unique locally convex domain. It is tempting for an optimisation practitioner to use the size of

[1]not necessarily total function evaluations since starting points closer to the local minima may provide better performance for a given local minimisation routines

the set of minimisers $|\mathcal{M}^3|$ as a stopping criterion for iterative sampling N of one dimensional objective functions. The practical usefulness of this idea can be demonstrated with the following example:

Example 2 The following instance of the optimisation problem has 13 local minima in the given domain

$$\min_x f(x) = -x\sin(x), \ x \in \Omega = [1, 80] \tag{3.4}$$

From Figure 3.4 we can deduce that the minimum number of sampling points required for $k = 3$ to find all local minima using the Sobol sequence is $N = 40$, this sampling is shown in Figure 3.5. If $N < 40$ then there aren't enough sampling points to deduce that there are at least 13 locally convex domains from using the same arguments as in Example 1. Note for example that if we used a sequence that skipped p_1 then $N = 39$ would be adequate since $l = 1 < p_{32} < p_{33}$. Using our Python implementation of TGO Endres (2016–b) with $N = 40$ all 13 local minima of the objective function were found in a total of 285 function evaluations.

An example of a stopping criterion would be to stop sampling if $|\mathcal{M}^3|$ is unchanged after, say, 10 sampling point evaluations. The rate at which the number of elements in $|\mathcal{M}^3|$ grows with increasing N also provides a heuristic for characterising the multimodality and the geometry of the objective function. Objective functions that have a large number of local minima in a small domain (and relatively fewer minima in other larger domains) will have a much smaller growth in $|\mathcal{M}^3|$ for a given low-discrepancy sampling. This idea of continuously classifying and extracting approximate function characteristic information from the sampling points will be formalised and extended to higher dimensions in chapter 4.

There is a simple reason why the 3-t-matrix has this quality in the first dimension for the optimisation problem given in Equation (3.1). However, it is not guaranteed that this property holds for any sampling point distribution. In fact it holds true only under the following conditions:

1. Consider all points in the ordered sampling set from the smallest to greatest x value $\mathcal{P} = \{p_i \mid p_0 \ < \ p_1 \ < \ p_2 \ \ldots \ < \ p_N - 1, \ p_i \ \in \ (x_l, \ x_u)\}$, excluding the supremum and infimum.

2. For any given point p_i the Euclidean distance between p_i and 2 of its nearest sampling points $p_{i-1} < p_i < p_{i+1}$ should be less than the relative difference between p_i and a fourth point in the sampling sequence $|p_i - p_j|$ where $j \neq i, i-1, i+1$.

In fact it is easy to prove both that for a locally, strictly convex domain of f the $3-$topograph construction can produce a larger minimiser pool $\mathcal{M}^3$ than optimal. It can also be shown that a construction must exist where the optimal number of minimisers will

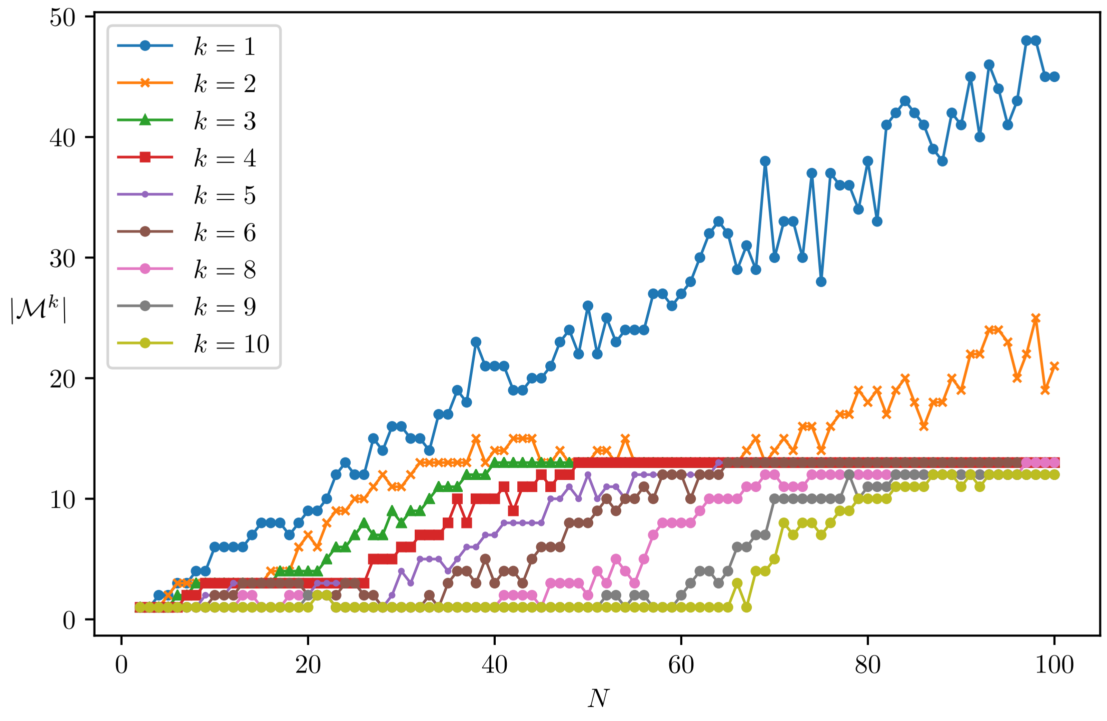

Figure 3.4: Number of minimisers $|\mathcal{M}^k|$ found using the TGO method for the given k values at various sampling points N

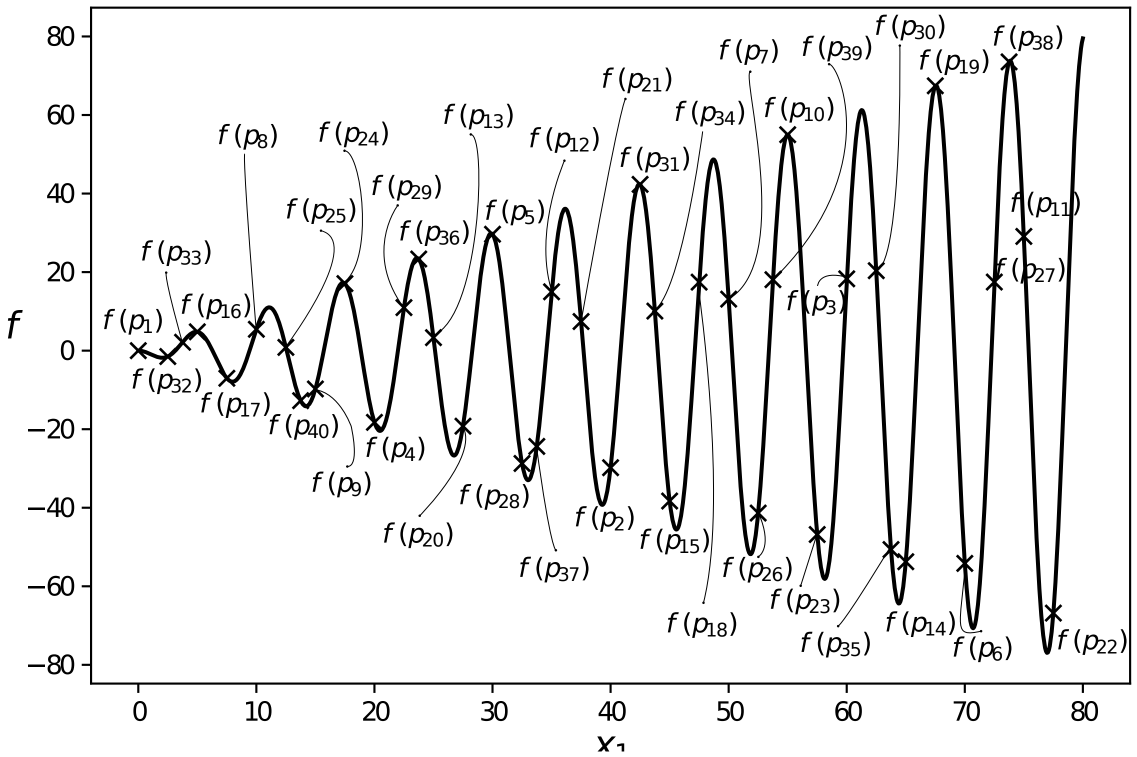

Figure 3.5: Plot of the objective function in Example 2 for $N = 40$ sampling points

always be extracted regardless of the sampling distribution. Furthermore it can be shown that at most 3 sampling points within a locally convex domain $x \in [x_l, x_u]$ is required to produce enough information so that only one minimiser in the domain is produced.

Theorem 1. *There exists a* 1*-dimensional sampling sequence such that* $k = 3$ *will produce a minimiser pool larger than optimal as defined by Definition 9.*

Proof. Consider a subdomain $x \in [x_l, x_u] \subset \mathbb{R}$ for which f is strictly convex. We define the set of N sampling points $\mathcal{P}$ ordered in such a way that

$$\mathcal{P} = \{p_i \mid p_0 \ < \ p_1 \ < \ p_2 < \ \ldots \ < \ p_{N-1}, \ p_i \ \in \ (x_l, \ x_u)\}$$

Let $\mathcal{F} = \{f_0, \ f_1, \ f_2, \ldots, \ f_{N-1}\}$ be set of one-to-one function values corresponding to the points mapped by $f : \mathcal{P} \to \mathcal{F}$.

Suppose we have $f_1 < f_0$ and $f_1 < f_2 < f_3, \ldots f_{N-1}$. By construction we have $|p_1 - p_2| < |p_1 - p_3| < |p_1 - p_4| < |p_1 - p_5|$ then by the Definitions 4, 5 and 6 p_2 is a minimiser of the $3 - t-$topograph. Suppose we have a sampling distribution such that $|p_2 - p_3| < |p_1 - p_2|, |p_2 - p_4| < |p_1 - p_2|$ and $|p_2 - p_5| < |p_1 - p_2|$ then by the definitions 4, 5 and 6 p_3 is also a minimiser of the $3 - t-$topograph. Therefore more than two minimisers are produced in the same locally convex sub-domain of $[x_l, x_u]$. We have shown that $\mathcal{M}^3$ can produce a minimiser pool larger than optimal which concludes the proof.

Lemma 1. *A construction exists that will always produce a minimiser pool larger than optimal as defined by Definition 9 for any given* 1*-dimensional sampling sequence.*

Now suppose that instead of using only the Euclidean distance metric we also invoke knowledge of the nearest point in every cartesian direction. We use the criterion that a minimiser point p_i is a minimiser iff with the ordering constructed in $\mathcal{P}$ and $\mathcal{F}$ we have $f_i < f_{i-1}$ and $f_i < f_{i+1}$. With this definition if the point p_i is a minimiser then no other point meets the criterion since by construction of the sampling in the locally convex domain $f_0 > f_1 > \cdots > f_{i-1} > f_i$ and $f_{i+1} < f_{i+2} < f_{i+3} < \cdots < f_{N-1}$. This proves Lemma 1.

Finally note that only information from the 3 points in the locally convex sub-domain of $[x_l, x_u]$ and their corresponding function values f_{i-1}, f_i and f_{i+1} are needed to produce a minimiser using this criterion. □

An important consequence here is that for low discrepancy sequences in higher dimensions and for less well behaved objective functions the topographs connected with the Euclidean distance metrics will similarly discard available information about the local geometry. This produces larger than optimal minimiser pools leading to very high numbers of function evaluations needed to solve the problem.

In the following section we will develop a more efficient algorithm that will make use of this information. SHGO will always produce equivalent results to this algorithm in the one dimensional case.

3.1 Axially directed topograph

Based on the observations from chapter 3 we develop an algorithm that, for a given sampling set, always uses the optimal number of starting minimisers as defined for one dimensional objective functions without requiring *a-priori* specification of the k parameter. Here a new graph structure is proposed and attempts are made to directly extend the idea to higher dimensions by connecting every vertex to the nearest vertex in every cartesian axis direction. In Theorem 2 we show that the one dimensional properties of this algorithm does not extend to higher dimensions which finally leads us to the built up complexes in chapter 4. The main conclusion of this section is that simpler graph structures cannot be used to find locally convex sub-domains of a function in the same way that was accomplished in chapter 3.

The algorithm proceeds in the same way as TGO described in chapter 2 except for step 2 where a new structure described in Section 4.1 replaces the topograph.

3.1.1 Axially directed topograph

Let $\mathcal{F}$ be the set of scalar outputs mapped by the objective function $f : \mathcal{P} \to \mathcal{F}$ for a given sampling set $\mathcal{P} \subseteq \Omega \subseteq \mathbb{R}^n$. The scalar elements $f_i \in \mathcal{F}$ have one-to-one correspondence with the vector elements $\mathbf{p}^i \in \mathcal{P}$ where the integer $i \in \{1, 2, 3, \ldots, N\}$ indicates the sampling point index. The vector $\mathbf{p}^i$ in turn has dimensional elements x^i_j where the integer $j \in \{1, 2, 3, \ldots, n\}$ indicates the dimension of the scalar value $\forall i(x^i_1,\ x^i_2,\ x^i_3,\ \ldots,\ x^i_n) \in \mathbf{p}^i$.

We wish to construct a graph that is ordered along the coordinate axes, this is done by formally defining the following related partially ordered sets.

Definition 10. *Given a finite structured set of N feasible ordered sampling points $\mathcal{P} = (\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^N)$ with its corresponding objective function outputs $\mathcal{F} = (f^1, f^2, \ldots, f^N)$, the index set of $\mathcal{P}$ is given as the ordered set $\mathcal{I} = (i = \{1, 2, 3, \ldots, N\}, \leq)$*

Note that the initial ordering of the index set is arbitrary, what's important is that an ordered index set is defined. This ordering will allow us to keep track of any vertex in the graph to its corresponding sampling point in $\mathcal{P}$ so that the corresponding objective function only needs to be evaluated once. Herein the order is taken as the order that is generated by the Sobol sequence.

Definition 11. *Given a set of feasible sampling points $\mathcal{P} \subseteq \Omega \subseteq \mathbb{R}^n$ define X_j for every dimension $j \in \{1, 2, 3, \ldots, n\}$ as the partially ordered set $X_j = \{\mathbf{p}^i \mid \forall i(x^i_j < x^{i+1}_j)\}$.*

The definition is demonstrated with the following numerical example:

Example 3 Given set of the first 5 points in the 2-dimensional Sobol sequence bounded by the 2-cube:

$$\mathcal{P} = ((0,\ 0),\ (0.5,\ 0.5),\ (0.75,\ 0.25),\ (0.25,\ 0.75),\ (0.375,\ 0.375)\ ) \subseteq [0,\ 1] \times [0,\ 1] \subseteq \mathbb{R}^2$$

let $f(x) = x_1^2 + x_2^2$ so that

$$\mathcal{F} = (0,\ 0.5,\ 0.625,\ 0.625,\ 0.28125)$$

then

$$X_1 = ((0,\ 0), (0.25,\ 0.75), (0.375,\ 0.375), (0.5,\ 0.5), (0.75,\ 0.25))$$

and

$$X_2 = ((0,\ 0), (0.75,\ 0.25), (0.375,\ 0.375), (0.5,\ 0.5), (0.25,\ 0.75))$$

The corresponding index sets are $\mathcal{I}_1 = (1, 4, 5, 2, 3)$ and $\mathcal{I}_2 = (1, 3, 5, 2, 4)$.

Definition 12. *For every dimension j, $\mathcal{F}_j$ is the partially ordered set such that the position of the elements X_j correspond to the original index sampling of $\mathcal{P}$, $\mathcal{F}_j = \{f_j^{i,k} \mid \forall i(x_j^i < x_j^{i+1}), f_j^{i,k} = f_k \in \mathcal{F}, k \subseteq \mathcal{I}\}$*

That is the first superscript i of the elements $f^{i,k}$ indicate the ordering in $\mathcal{F}_j$, while the second superscript k indicates the corresponding scalar value of $f^{i,k}$ in $\mathcal{F}$. Ordering the example we have $\mathcal{F}_1 = (0,\ 0.625,\ 0.28125,\ 0.5,\ 0.625)$ and $\mathcal{F}_2 = (0,\ 0.625,\ 0.28125,\ 0.5,\ 0.625)$.

Definition 13. *For every dimension j, define the partially ordered sets of cardinality N such that $\mathcal{F}_j^+ = \{f_j^{i,k} - f_j^{i-1,k} \mid \forall i(x_j^i < x_j^{i+1}), f_j^{i,k} = f_k \in \mathcal{F}, i = \{1, 2, \ldots, N, k \subset \mathcal{I}\}\}$ and $\mathcal{F}_j^- = \{f_j^{i,k} - f_j^{i+1,k} \mid \forall i(x_j^i < x_j^{i+1}), f_j^{i,k} = f_k \in \mathcal{F}, i = \{0, 1, \ldots, N-1\}, k \subset \mathcal{I}\}$*

These sets are essentially objective function differences between the sampling points along each dimensional Cartesesian axis. Continuing from the numerical example we have

$$\begin{aligned}
\mathcal{F}_1^+ &= (\ 0.625, -0.34375,\ 0.21875,\ 0.125) \\
\mathcal{F}_2^+ &= (-0.625,\ 0.34375, -0.21875, -0.125) \\
\mathcal{F}_1^- &= (\ 0.625, -0.34375,\ 0.21875,\ 0.125) \\
\mathcal{F}_2^- &= (-0.625,\ 0.34375, -0.21875, -0.125)
\end{aligned}$$

We denote the elements as $f_j^{+i,k} \in \mathcal{F}_j^+$ and $f_j^{-i,k} \in \mathcal{F}_j^-$ for every dimension $j \in \{1, 2, 3, \ldots, n\}$, cartesian ordering $i \subseteq \mathcal{I}$ and corresponding sampling point $k \in \mathcal{I}$. The usefulness of these abstract constructions is apparent in the following definition.

Definition 14. *For a given sampling set $\mathcal{P}$. The minimiser pool $\mathcal{M}$ is defined as*

$$\mathcal{M} = \mathcal{M}_c \cup \mathcal{M}_{lb} \cup \mathcal{M}_{ub}$$

where

$\mathcal{M}_c = \left\{ \mathbf{p}^i \mid \forall j \left((f_j^{+i} > 0) \wedge (f_j^{-(i+1)} > 0) \right), i = \{1, 2, 3, \ldots, N-1\} \right\}$
$\mathcal{M}_{lb} = \left\{ \mathbf{p}^i \mid \forall j \left(f_j^{-i} < 0 \right), i = \{0\} \right\}$
$\mathcal{M}_{ub} = \left\{ \mathbf{p}^i \mid \forall j \left(f_j^{+i} < 0 \right), i = \{N\} \right\}$

That is, we simply check the finite difference between sampling points in every cartesian direction. In addition we check if the sampling points on the boundaries are minimisers.

Theorem 2. *The minimiser pool $\mathcal{M}$ from Definition 14 always produces a set that is either smaller than or equal to the optimum minimiser pool as defined by Definition 9 iff $j = 1$.*

Proof. The proof for $j = 1$ follows the same argument from chapter 3. By Definition 10, 11 and 12 we have the ordering constructed as $\mathcal{P}$ and $\mathcal{F}_1$. If a given point $\mathbf{p}^i$ is a minimiser with $f_1^{+i} > 0$ and $f_1^{-i} > 0$, then we have by Definition 13 $f^i < f^{i-1}$ and $f^i < f^{i+1}$, conversely if a given point $\mathbf{p}^i$ is not a minimiser then either $f_1^{+i} < 0$ or $f_1^{-i} > 0$ so that regardless of the sampling method used and the Euclidean distance between points a minimiser will never be generated for any point that has $((f^i > f^{i-1}) \wedge (f^i > f^{i+1})) \vee ((f^i < f^{i-1}) \wedge (f^i < f^{i+1}))$.

If $j > 1$ we have no such guarantee for a higher dimensional locally convex domain. As a counter example consider the set of points

$$\mathcal{P} = ((0,\ 0), (0.25,\ 0.25), (0.75,\ 0.125), (0.125,\ 0.75))$$

on the same function as above, the minimiser set produced is $\mathcal{M} = \{(0,\ 0),\ (0.25,\ 0.25)\}$ which is clearly larger than optimal and will produce the same local minimum. $\square$

This unsatisfactory result for higher dimensions could still potentially show good performance for more regular spaced sampling such as grids, however, as we will see in the next section the SHGO algorithm can guarantee that the optimal minimiser set will be produced for any dimension.

3.1.2 Implementation

Algorithm 1 provides a high-level overview of the ATGO algorithm. A Python implementation of this algorithm can be found in Endres (2016–a).

Algorithm 1 ATGO algorithm

```
1: procedure INITIALISATION
2:     Input an objective function $f$, constraint functions $\mathbf{g}$ and variable bounds and $[\mathbf{l}, \mathbf{u}]^n$.
3:     Input $N$ initial sampling points.
4:     Define a sampling sequence that generates a set $\mathcal{X}$ of sampling points in the unit hypercube space $[\mathbf{0}, \mathbf{1}]^n$
5: end procedure
6: procedure INITIAL SAMPLING
7:     $\mathcal{P} = \emptyset$
8:     while $|\mathcal{P}| < N$ do
9:         Generate $N - |\mathcal{P}|$ sequential sampling points $\mathcal{X} \subset \mathbb{R}^n$
10:        Stretch $\mathcal{X}$ over the lower and upper bounds $[\mathbf{l}, \mathbf{u}]^n$
11:        $\mathcal{P} = \{\mathcal{X}_i \mid \mathbf{g}(\mathcal{X}_i) \geq 0, \forall \mathcal{X}_i \in \mathcal{X}\} \cup \mathcal{P}$    ▷ (Find $\mathcal{P}$ in the feasible subset $\Omega$ by discarding any points mapped outside the linear constraints $g$ and adding to the current set of $\mathcal{P}$.)
12:        Set $\mathcal{X} = \emptyset$
13:    end while
14:    Find $\mathcal{F}$ from the objective function $f : \mathcal{P} \to \mathcal{F}$
15: end procedure
16: procedure CONSTRUCT $\mathcal{M}$
17:    Calculate $\mathcal{M}$ from the sets $\mathcal{P}$ and $\mathcal{F}$ using Definitions 11 through 14.
18: end procedure
19: procedure LOCAL MINIMISATION
20:    Calculate the approximate local minima of $f$ using a local minimisation routine with the elements of $\mathcal{M}$ as starting points.    ▷ These local minimisations can be performed in parallel.
21: end procedure
22: procedure PROCESS RETURN OBJECTS
23:    Order the final outputs of the minima of $f$ found in the local minimisation step to find the approximate global minimum.
24: end procedure
25:
26: return the approximate global minimum and a list of all the minima found in the local minimisation step.
```

CHAPTER 4

Simplicial Homology Global Optimisation

4.1 Overview

The SHGO method strongly relies on constructing a simplicial complex using the sampled points of an objective function f as vertices. From this construction of the complex $\mathcal{H}$ we use the resulting directed subgraph which contains the set of all $1-$chains from the elements of $\mathcal{H}^1 \in \mathcal{H}$ to find minimiser pools using definitions similar to the methods demonstrated in the previous sections. This is accomplished by the application of Sperner's lemma Sperner (1928) allowing us to approximate the domains of stationary points for any objective function in the feasible search space Ω.

We prove that, if provided with an adequate sampling set, the construction of $\mathcal{H}$ will produce the same homology groups. We use this result to show that for the given sampling set of vertices $\mathcal{H}^0 \in \mathcal{H}$ we always extract the optimal minimiser pool similar to the one-dimensional case described in chapter 3, but extended to higher dimensions.

The algorithm itself consists of four steps which will described in detail:

1. Uniform sampling point generation of N vertices in the search space within the bounded and constrained subspace of Ω from which the $0-$chains of $\mathcal{H}^0$ are constructed.

2. Construction of the directed simplicial complex $\mathcal{H}$ by triangulation of the vertices.

3. Construction of the minimiser pool $\mathcal{M} \subset \mathcal{H}^0$ by repeated application of Sperner's lemma.

4. Local minimisation using the starting points defined in $\mathcal{M}$.

We will start by formally defining the construction of $\mathcal{H}$ from a given set of feasible sampling points $\mathcal{P}$ and proving its properties.

4.2 Directed simplicial complex approximation of the objective function

Consider again the general objective function mapping in the continuous domain $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The purpose of this section is describe a discrete mapping $h : \mathcal{P} \rightarrow \mathcal{H}$ to provide a simplicial approximation for the surface of f. To guide the reader the methods will be demonstrated on the simple 2-dimensional optimisation problem defined in Example 4. The use of a 2-dimensional surface allows a demonstration of the techniques while the abstractions defined are readily extended to higher dimensions.

We start by formally defining the set of vertices from which $0-$chains of the simplicial complex are built and the of edges from which the $1-$chains of $\mathcal{H}$ are built.

Definition 15. *Let $\mathcal{X}$ be the set of sampling points generated by a sampling sequence in the bounded hyperrectangle $[\mathbf{l}, \mathbf{u}]^n$. The set $\mathcal{P} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{g}(\mathbf{x}) \geq 0\}$ is a set of points within the feasible set Ω .*

Definition 16. *For an objective function f, $\mathcal{F}$ is the set of scalar outputs mapped by the objective function $f : \mathcal{P} \rightarrow \mathcal{F}$ for a given sampling set $\mathcal{P} \subseteq \Omega \subseteq \mathbb{R}^n$.*

Definition 17. *Let $\mathcal{H}$ be a directed simplicial complex. Then $\mathcal{H}^0 := \mathcal{P}$ is the set of all vertices of $\mathcal{H}$.*

Definition 18. *For a given set of vertices $\mathcal{H}^0$, the simplicial complex $\mathcal{H}$ is constructed by a triangulation connecting every vertex in $\mathcal{H}^0$. The triangulation supplies a set of undirected edges E.*

Definition 19. *The set $\mathcal{H}^1$ is constructed by directing every edge in E. A vertex $v_i \in \mathcal{H}^0$ is the connected to another vertex v_j by an edge contained in E. The edge is directed as $\overline{v_i v_j}$ from v_i to v_j iff $f(v_i) < f(v_j)$ so that $\partial\left(\overline{v_i v_j}\right) = v_j - v_i$. Similarly an edge is directed as $\overline{v_j v_i}$ from v_j to v_i iff $f(v_i) > f(v_j)$ so that $\partial\left(\overline{v_j v_i}\right) = v_i - v_j$.*

For practical computational reasons we must also consider the case where $f(v_i) = f(v_j)$. If neither v_i or v_j is already a minimiser we will make use of rule that the incidence direction of the connecting edge is always directed towards the vertex that was generated earliest by the sampling point sequence. If v_i is not connected to another vertex v_k then we leave the notation $\overline{v_i v_k}$ undefined and let $\partial\left(\overline{v_i v_k}\right) = 0$. We let the higher dimensional simplices of $\mathcal{H}^k, k = 2, 3, \ldots n+1$ be directed in any arbitrary direction which completes the construction of the complex $h : \mathcal{P} \rightarrow \mathcal{H}$. We can now use $\mathcal{H}$ to find the minimiser pool for the local minimisation starting points used by the algorithm:

Definition 20. *A vertex v_i is a minimiser iff every edge connected to v_i is directed away from v_i, that is $\partial\left(\overline{v_i v_j}\right) = (v_{j \neq i} - v_i) \vee 0 \ \forall v_{j \neq i} \in \mathcal{H}^0$. The minimiser pool $\mathcal{M}$ is the set of all minimisers.*

We will also make extensive use of star notation Hatcher (2002); Henle (1979):

Definition 21. *The star of a vertex v_i, written $st(v_i)$, is the set of points Q such that every simplex containing Q contains v_i.*

The $k-$chain $C(\mathcal{H}^k), k = n + 1$ of simplices in $\text{st}\,(v_i)$ forms a boundary cycle $\partial(C(\mathcal{H}^{n+1}))$ with $\partial\,(\partial(C(\mathcal{H}^{n+1}))) = \emptyset$. The faces of $\partial(\mathcal{H}^{n+1})$ are the bounds of the domain defined by $\text{st}\,(v_i)$.

A visual demonstration of these constructions and notations is provided in the following numerical example:

Example 4 The Ursem01 function for two dimensions is defined as follows Gavana (2016)

$$\min f(\mathbf{x}) = -\sin{(2x_1 - 0.5\pi)} - 3\cos{(x_2)} - 0.5x_1,\ x \in \Omega = [0, 9] \times [-2.5, 2.5]$$

Figure 4.1 provides a 3 dimensional plot of this function. The function has three local minima within the domain $\mathbf{x} \in [0, 9] \times [-2.5, 2.5]$.

We use a set $\mathcal{P}$ of 15 sampling points from the 2-dimensional Sobol sequence. First map out the objective function values:

$$f : \begin{bmatrix} v_0 = (0.0, -2.5) \\ v_1 = (4.6, 0.0) \\ v_2 = (6.9, -1.25) \\ v_3 = (2.3, 1.25) \\ v_4 = (3.45, -0.625) \\ v_5 = (8.05, 1.875) \\ v_6 = (5.75, -1.875) \\ v_7 = (1.15, 0.625) \\ v_8 = (1.725, -0.9375) \\ v_9 = (6.325, 1.5625) \\ v_{10} = (8.625, -2.1875) \\ v_{11} = (4.025, 0.3125) \\ v_{12} = (2.875, -1.5625) \\ v_{13} = (7.475, 0.9375) \\ v_{14} = (5.175, -0.3125) \end{bmatrix} \rightarrow \begin{bmatrix} f_0 = 3.403 \\ f_1 = -6.275 \\ f_2 = -4.0651 \\ f_3 = -2.208 \\ f_4 = -3.3429 \\ f_5 = -4.051 \\ f_6 = -1.493 \\ f_7 = -3.674 \\ f_8 = -3.591 \\ f_9 = -2.191 \\ f_{10} = -2.606 \\ f_{11} = -5.062 \\ f_{12} = -0.601 \\ f_{13} = -6.239 \\ f_{14} = -6.044 \end{bmatrix} \tag{4.1}$$

From Definition 17 we find $\mathcal{H}^0$ from $\mathcal{P}$. Next we use Delaunay triangulation to find a set of connected edges according to Definition 18. Any triangulation scheme resulting in a simplicial complex can be used. Next the edges are directed from the calculated values of $\mathcal{F}$ using Definition 19. Finally from Definition 20 we find the minimiser set $\mathcal{M} =$

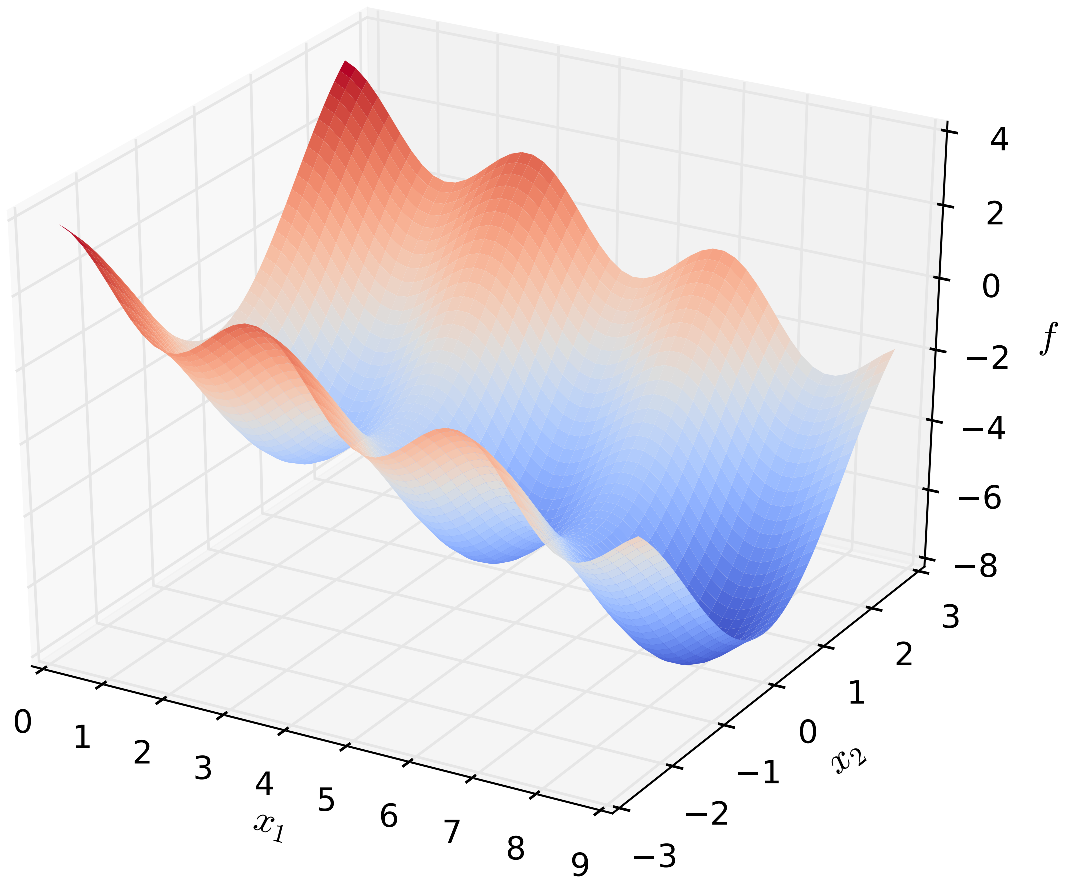

Figure 4.1: A 3-dimensional surface plot of the optimisation test function given in Example 4 $f(\mathbf{x}) = -\sin(2x_1 - 0.5\pi) - 3\cos(x_2) - 0.5x_1$ for the domain $\mathbf{x} \in \Omega = [0, 9] \times [-2.5, 2.5]$

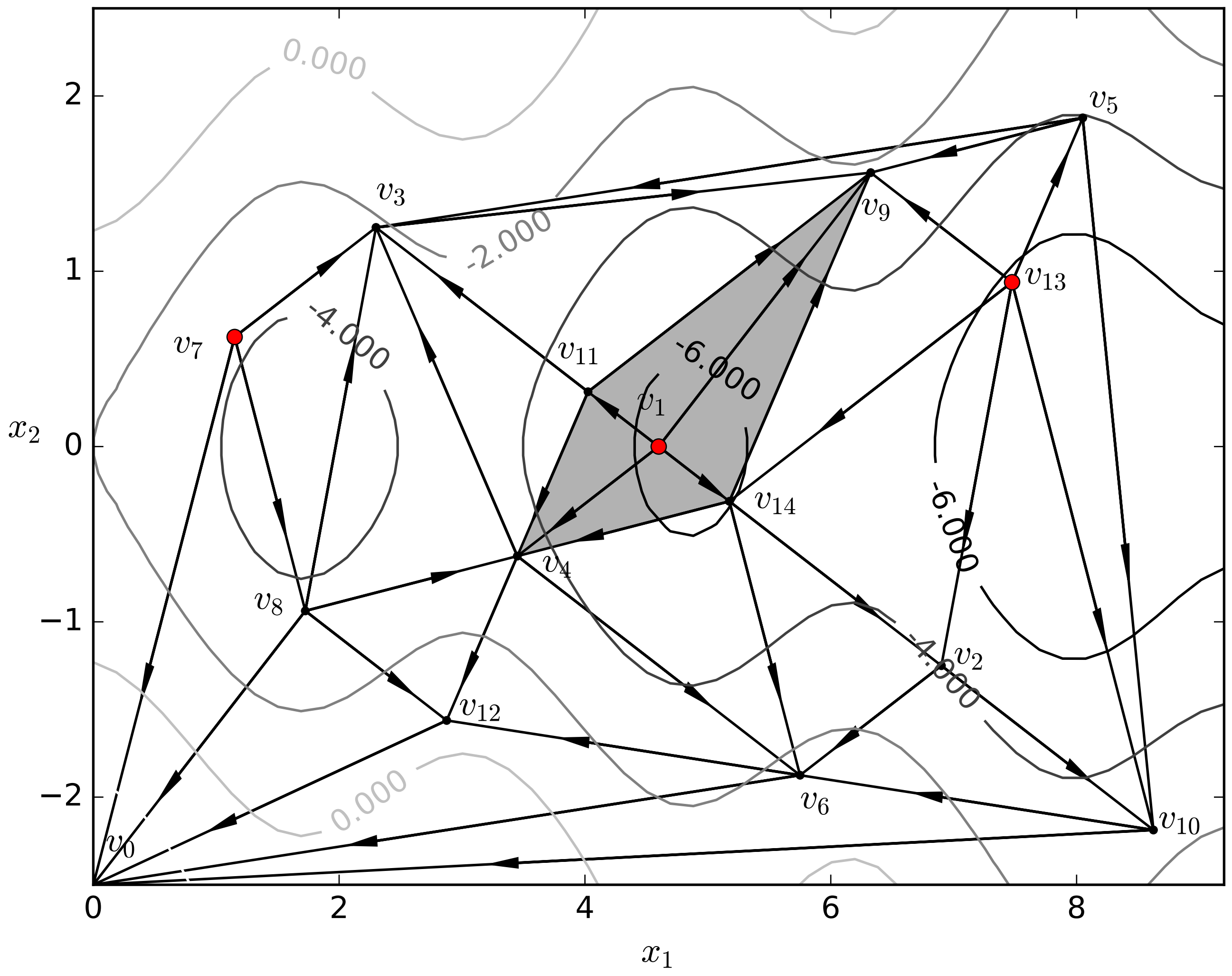

Figure 4.2: A directed complex $\mathcal{H}$ forming a simplicial approximation for an objective function. There are three minimiser vertices v_1, v_7 and v_{13} shown by the big red dots. The area shaded in grey represents the domain defined by $\mathrm{st}\,(v_1)$

$\{v_1, v_7, v_{13}\}$. The resulting structure is shown in Figure 4.2. Also shown in Figure 4.2 is the domain of $\mathrm{st}\,(v_1)$ for a visual description of Definition 21. Next we increase the sampling size to $N = 150$ points and repeat the procedure. The resulting complex is shown in Figure 4.3. Notice that while the minimiser vertices have changed (now closer to the true continuous local minima), the cardinality of the minimiser pool $|\mathcal{M}|$ remains unchanged. That is, given an adequate number sampling points $|\mathcal{M}|$ will cease to grow with increasing N, providing a heuristic for the number of sampling points needed to approximately map all minima of an objective function. This useful property of the SHGO algorithm is proven formally in section 4.4.

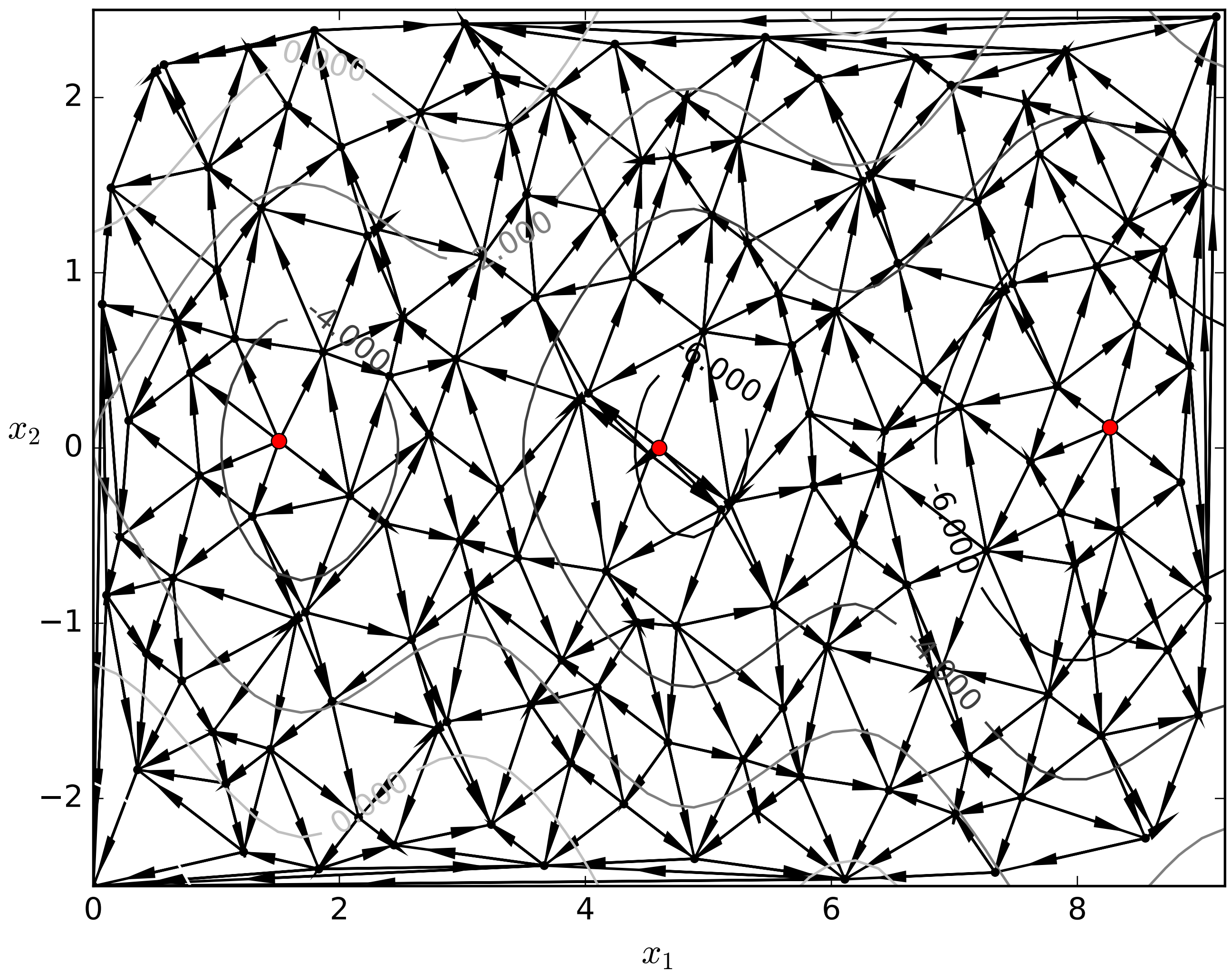

Figure 4.3: A directed complex $\mathcal{H}$ forming a simplicial approximation for an objective function with 150 vertices. There are three minimiser vertices given by the big red dots

4.3 Guarantee of stationary points in sub-domains near minimiser points

This section is devoted to proving the following theorem:

Theorem 3. *Given a minimiser $v_i \in \mathcal{M} \subseteq \mathcal{H}^0$ on the surface of a continuous, Lipschitz smooth objective function f with a compact bounded domain in $\mathbb{R}^n$ and range $\mathbb{R}$, there exists at least one stationary point of f within the domain defined by $st(v_i)$.*

Proof. Our strategy relies on finding a simplex with a Sperner labelling where each label represents a different $n+1$ label in every vector direction of the gradient vector field ∇f of f where of the $n+1$ Cartesian directions we require only a vector pointing towards a section defined by $n+1$ hyperplane cuts, the remainder of the proof then proceeds as usual for Brouwer's fixed point theorem Brouwer (1911) found in for example Henle (1979: p. 40) utilising Sperner's lemma.

Theorem 4. *(Sperner's lemma (Sperner, 1928)) Every Sperner labelling of a triangulation of a n-dimensional simplex contains a cell labelled with a complete set of labels: 1,2, . . . , n+1.*

Start with the observation that for any minimiser $v_i \in \mathcal{M} \subseteq \mathcal{H}^0$ we have by construction that for any vertex v_j with incidence on a connecting edge $\overline{v_i v_j}$ that $f(v_i) < f(v_j)$, so by the MVT there is at least one point on $\overline{v_i v_j}$ where ∇f points towards a Cartesian direction in a section that can receive a unique Sperner label. If we have $n+1$ vertices with incidence on an edge $\overline{v_i v_j} \subseteq \mathcal{H}^1$ in every required Cartesian direction then we have a simplex within $\mathrm{st}\,(v_i)$ with a Sperner labelling.

In the case where we do not have $n+1$ vertices in every required section then by construction there is no vertex between v_i and the boundary of f defined by Ω in the required section. In the case where the constraint is not active and there exists at least one point v_k boundary where ∇f does not point towards the boundary and by the MVT v_k can receive a unique Sperner label from which we can construct a simplex within $\mathrm{st}\,(v_i)$ with Sperner labelling.

Following the combinatorial version of Brouwer's fixed point theorem Henle (1979) since ∇f is continuous and the domain $\mathrm{st}\,(v_i)$ is compact we can produce a sequence of complete triangulations with arbitrarily small size in which the size of the simplices decreases toward zero. This sequence produces a sequence of vertices with gradients $\nabla f(V)$ pointing in every $n+1$ direction. By continuity there is a vector $\nabla f(\mathbf{X})$ near the sequences, since the zero vector is the only vector pointing in all $n+1$ directions we have a point $\mathbf{X}$ bounded by the domain defined by $\mathrm{st}\,(v_i)$ where $\nabla f(\mathbf{X}) = \bar{0}$. In the case where the constraint is active a local minimum lies on the constraint which is in the domain defined $\mathrm{st}\,(v_i)$. This concludes the proof. □

Figure 4.4 provides a visual demonstration of the proof using the complex from Example 4. Here we have divided the plane so that the 3 required directions are $[0, \frac{\pi}{2})$, $[\frac{\pi}{2}, \pi)$ and $[\pi, 2\pi)$. Note that this division is arbitrary and any $n+1 = 3$ subdivisions can be chosen as long as all possible $n+1 = 3$ directions can form a simplex in the space are covered. The three possible simplices are contained within the star domains of each minimiser $\mathrm{st}\,(v_1)$, $\mathrm{st}\,(v_7)$ and $\mathrm{st}\,(v_{13})$.

First consider the minimiser v_{13}. There are three possible edges in $[\frac{\pi}{2}, \pi)$ on which a point exists that can be used as a vertex to receive a Sperner labelling for that direction namely $\overline{v_{13}v_{14}}$, $\overline{v_{13}v_2}$ and $\overline{v_{13}v_{10}}$. The only possible edges in the $[0, \frac{\pi}{2})$, $[\frac{\pi}{2}, \pi)$ directions are $\overline{v_{13}v_5}$ and $\overline{v_{13}v_9}$ respectively. The simplex $\overline{v_5v_9v_{10}}$ drawn in Figure 4.4 is not necessarily the simplex with a Sperner labelling. The three vertices of the Sperner simplex which are proven to exist through the MVT exists on each of the edges $\overline{v_{13}v_{14}}$, $\overline{v_{13}v_2}$ and $\overline{v_{13}v_{10}}$ in a subdomain of this simplex $\overline{v_5v_9v_{10}}$. For example the simplex surrounding the minimiser v_1 is a possible Sperner simplex with vertices on the edges in every required direction.

Note that if the edge $\overline{v_{13}v_{14}}$ was chosen instead of $\overline{v_{13}v_{10}}$ then the local minimum of the function would be outside the domain of the simplex with the Sperner labelling. This is an important observation because it demonstrates that Theorem 3 cannot be used to further refine the location of the local minimum from the domain $\mathrm{st}\,(v_{13})$ using mechanisms of the proof, it only states that at least one local minimum exists within $\mathrm{st}\,(v_{13})$.

The boundaries of $\mathrm{st}\,(v_{13})$ can be found using the $3-$chain $C_{13}(\mathcal{H}^3)$ of simplices in $\mathrm{st}\,(v_{13})$, recall that the directions of simplices higher than dimension 2 are undefined and so the directions can be arbitrarily chosen

$$C_{13}(\mathcal{H}^3) = \overline{v_{13}v_{10}v_5} + \overline{v_{13}v_5v_9} + \overline{v_{13}v_9v_{14}} + \overline{v_{13}v_{14}v_2} + \overline{v_{13}v_2v_{10}}$$

$C_{13}(\mathcal{H}^3)$ clearly forms a cycle, applying the boundary operator we find the faces defining the bounds of the domain of $\mathrm{st}\,(v_i)$ which in this case is the chain of edges with defined direction

$$\partial(C_{13}(\mathcal{H}^3)) = -\overline{v_{10}v_5} + \overline{v_5v_9} - \overline{v_9v_{14}} + \overline{v_{14}v_2} + \overline{v_2v_{10}}$$

thus $\partial\left(\partial(C(\mathcal{H}^3))\right) = \emptyset$.

$v_7 = (1.15, 0.625)$ is an example of a minimiser that does not have all three required directions for a Sperner labelling, the light red shaded area represents the area wherein a local minimum can exist. For example on the lines $x_1 = 0$ for $x_2 \in [0.625, 2.5]$ or $x_2 = 2.5$ for $x_1 \in [0, 1.15]$ there will either exist a point $\mathbf{p}$ where the gradient $\nabla f(\mathbf{p})$ points in any direction pointing towards $[\frac{3}{2}\pi, 0)$ in which case and edge $\overline{v_{13}\mathbf{p}}$ exists that points in the $[\frac{\pi}{2}, \pi)$ direction and we have a simplex with a Sperner labelling. For example the dotted

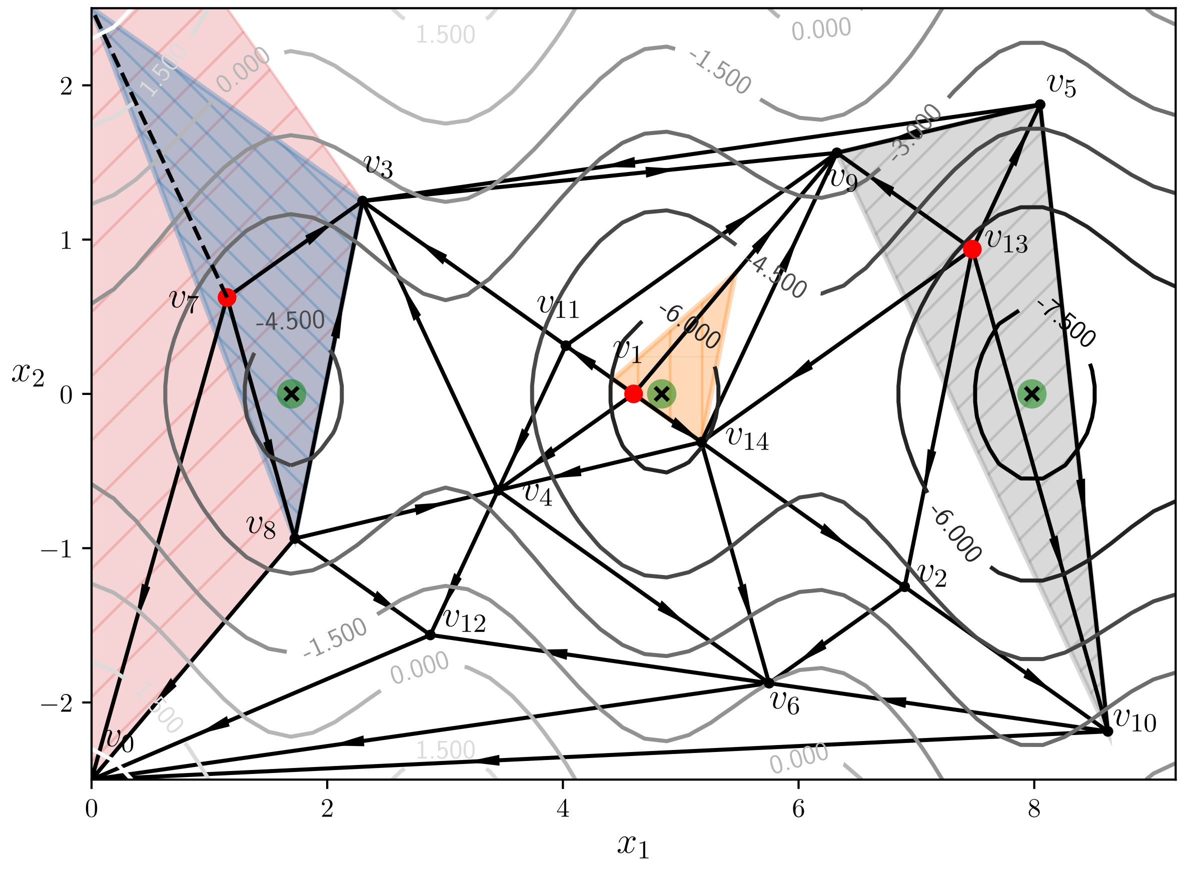

Figure 4.4: Visual demonstration of the proof by finding simplices with Sperner labellings. The three circled crosses are the (approximate) minimima of the objective function within the given bounds. The three possible Sperner simplices are contained within the star domains of each minimiser st (v_1), st (v_7) and st (v_{13}). v_7 is an example of simplices without complete Sperner labelings, the red shaded area around $\overline{v_7}$ is the bounded domain wherein at least one local minimum exist

line on Figure 4.4 with the Sperner simplex represented by blue shaded around v_7. If such a point does not exists then all points on those lines points in the $[0, \frac{3}{2}\pi)$ direction and so one or more local minimum lies somewhere on the boundary which is within the defined area.

There have been several developments in the extension of this lemma which could prove useful in applications extending the SHGO algorithm. One of the most interesting is by De Loera, Peterson, and Edward Su (2002) where they proved the Atanassov conjecture Atanassov (1996) that for any polytope with N vertices there are $N - n$ simplices that receive a complete set of Sperner labels. Meunier (2006) further extended this theorem and more recently Musin (2015) extended the theorems to a large class of manifolds with or without boundary. These theorems could prove useful for extending the algorithm to make use of this information. More explicitly, SHGO currently uses knowledge of the objective function evaluations, but only in a Boolean sense (in the form of directed edges). The theorems by Meunier and Musin allow us to extend Sperner's lemma to a simplicial complex built in a $(n + 1)$-dimensional non-euclidean space. This would allow the application of ideas from discrete differential geometry. For example the Gauss-Bonnet theorem holds for discrete simplicial surfaces Keenan Crane (2013). The Gauss-Bonnet theorem provides a relation between the total Gaussian curvature and the Euler characteristic of a surface. By simple summation of the angle defect around every vertex we can determine the Euler characteristic of a continuous surface. As will be demonstrated in Section 5.4 the simplicial complex used by SHGO is homeomorphic to complexes built on other topological hypersurfaces. Therefore when using explicit coordinates of the expected homomorphism the summation can be used to compare the error with the Euler characteristic which provides a metric for how accurately the objective function surface has been sampled. In global optimisation theory a simplicial complex built in this space can be used for approximating local and global Lipschitz constants for an objective function while still retaining the ability to detect locally convex sub-domains in the search space.

4.4 Invariance of the directed complex within a bounded rectangle

We now have a guarantee of finding stationary points in sub-domains near stationary points. However, we would also like to ensure that SHGO does not generate more than one minimiser starting point per convex sub-domain. This can only be guaranteed when an objective function surface is "adequately sampled". For black box functions there is no way to know if the number and distribution of sampling points is adequate without more information (for example if the number of local minima are known in the problem).

However, it is an important property of the algorithm that $|\mathcal{M}|$ will stop increasing with higher sampling after this point. First we define an adequately sampled surface.

Definition 22. *Consider a simplicial complex $\mathcal{H}$ built on an objective function f with a compact feasible set Ω using Definitions 17 through 20. The surface is said to be* **adequately sampled** if there is one and only one true stationary point within every domain defined by Theorem 3.

The remainder of this section is devoted to proving the following theorem which holds in the case where $\Omega = [\mathbf{l}, \mathbf{u}]^n$.

Theorem 5. *(Invariance of an adequately sampled simplicial complex $\mathcal{H}$) For a given continuous objective function f that is adequately sampled by a sampling set of size N. If the cardinality of the minimiser pool extracted from the directed simplex $\mathcal{H}$ is $|\mathcal{M}|$. Then any further increase of the sampling set N will not increase $|\mathcal{M}|$.*

Proof. The proof relies on a homomorphism between the simplicial complex $\mathcal{H}$ constructed in the bounded hyperrectangle Ω and the homology (mod 2) groups of a constructed surface $\mathcal{S}$ on which we can invoke the invariance theorem.

Define the n-torus $\mathcal{S}_0$ from the compact, bounded hyperrectangle Ω by identification of the opposite faces and all extreme vertices. Now for every strict local minimum point $\mathbf{p} \in \Omega$ puncture a hypersphere and after appropriate identification the resulting n-dimensional manifold $\mathcal{S}_g$ is a connected g sum of g tori $S := S_0 \# S_1 \# \cdots \# S_{g-1} \qquad (g \text{ times})$

Any triangulation $\mathcal{K}$ of the topological space $\mathcal{S}$ is homeomorphic to $\mathcal{S}$, $\mathbf{H}_k(\mathcal{K}) \cong \mathbf{H}_k(\mathcal{S})\ \forall k \subset \mathbb{Z}$. Note that this homomorphism is for a mod 2 homology between a triangulation $\mathcal{K}$ and the surface $\mathcal{S}$ and is thus undirected. A triangulation corresponding to all vertices and faces of $\mathcal{K}$ can be directed according to Definition 17, Definition 18 and Definition 19 providing the directed simplicial complex $\mathcal{H}$. By construction we have, for an adequately sampled simplicial complex $\mathcal{H}$, an equality which exists between the cardinality of $\mathcal{M}$ and the Betti numbers of $\mathcal{S}$ as $|\mathcal{M}| = h_1 = rank(\mathbf{H}_1(\mathcal{S})) = rank(\mathbf{H}_1(\mathcal{K}))$. Here we invoke the invariance theorem

Theorem 6. *(Invariance theoremHenle (1979)) The homology groups associated with a triangulation $\mathcal{K}$ of the a compact, connected surface $\mathcal{S}$ are independent of $\mathcal{K}$. In other words, the groups $\mathbf{H}_0(\mathcal{K})$, $\mathbf{H}_1(\mathcal{K})$ and $\mathbf{H}_2(\mathcal{K})$ do not depend on the simplices, incidence coefficients, or anything else arising from the choice of the particular triangulation $\mathcal{K}$; they depend only on the surface $\mathcal{S}$ itself.*

The invariance theorem can be extended to higher dimensional triangulable spaces using singular homology through the Eilenberg-Steenrod Axioms Eilenberg and Steenrod (1952); Henle (1979). As a direct consequence any triangulation of $\mathcal{S}$ will produce the same homology groups for $\mathcal{K}$.

Adding any new sampling point within the corresponding subdomains of $\mathrm{st}\,(v_i)\ \forall i(v_i \in \mathcal{M} \subseteq \mathcal{H}^0)$ as defined in Theorem 3 will by definitions 17 through 20 need to be connected directly to v_i by a new edge or the triangulation is no longer a simplicial complex and thus not increase $|\mathcal{M}|$ since only one vertex will be the new minimiser.

After adding any sampling point outside a domain $\mathrm{st}\,(v_i)$ then, through the established homomorphism, any construction of $\mathcal{H}$ will produce the same homology groups since $rank(\mathbf{H}_1(\mathcal{K}))$ remains unchanged and it is thus not possible for a new vertex to be wrongly identified as a minimiser in the triangulation $\mathcal{H}$.

This concludes the proof that any increase in N will not further increase $|\mathcal{M}|$. $\square$

It is important to note that Theorem 5 is only applicable to complexes with adequate sampling as defined, that is to say it is entirely possible that, in complexes with less that adequate sampling, two starting minimiser elements of $\mathcal{M}$ will converge to the same local minimum. This flaw is inherent in the fact that there is insufficient information to completely identify the minima of a surface (and could be overcome if some extra information about f is known).

Theorem 3 and Theorem 5 also lead to the following corollary about an optimisation problem:

Corollary 2. *Consider any objective function $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$. Consider also a local minimisation routine that is guaranteed to converge to a local minimum in the same locally convex domain as the starting point inputted to the algorithm. Alternatively the local minimisation routine is guaranteed to converge to a point within a set of bounds (provided by the boundary of the $k-$chain around $st(v_i)$, $\partial\left(C(\mathcal{H}^k)\right), k = n+1$). If such a local minimisation routine uses an element $v_i \in \mathcal{M}$ as a starting point and the routine leads to a minimum outside or on $st(v_i)$ and in addition the minimum is not contained in the set $\mathcal{H}^0$. Then it can be concluded that either search space is not adequately sampled or f is not a Lipschitz smooth function.*

Therefore according to 2 if the number of local minima are known, as in for example phase equilibria problems, then we can extract valuable information about the objective function. In particular it can be determined whether or not the objective function is Lipschitz smooth. Alternatively if the function is known to be Lipschitz smooth then 2 can be used to prove the sampling is insufficient when the condition is not met. When this happens it is also now known that there are more local minima to be found, one or more of which might possibly be the global minimum. 2 does not, however, provide any guarantee that the sampling is sufficient when the conditions are met.

4.5 Sampling generation

Using the Sobol sequence sampling point generation proceeds in a similar way as that described in section 2.1. However, rather than only generating an arbitrary number of pre-defined sampling points we will also consider heuristic methods starting with the minimum amount of sampling points required to triangulate an n dimensional space. For example start with the minimum amount of sampling points to construct an $n-$dimensional simplex and continue sampling while continuously calculating the $\mathbf{H}_1(\mathcal{H})$ homology groups of the complex. Using the definitions described in this section the sampling is continued until the growth rate of the approximated homology groups slows appreciably.

In this publication the Sobol sequenced sampling points are triangulated using Delaunay triangulation as implemented in the SciPy library Jones et al. (2001–). A major disadvantage to this triangulation scheme is that it does not scale well to higher dimensions since it relies on solving convex hull using the quickhull method developed by Barber and Dobkin (1996). There are several possibilities for mitigating this problem. Since the Sobol sequence is deterministic the triangulations can be calculated and stored in a database. For SHGO another possibility whereby the convex hull does not need to be solved by using symmetry generated triangulation was developed. Building on the initial n-cube triangulation developed by Paulavičius & Žilinskas Paulavičius & Žilinskas (2014a); Žilinskas (2008) and using the symmetry groups S_n, $n = \{1, 2, 3, \ldots,\text{n}\}$ to generate an initial triangulation. Subsequent uniform sampling that ensures a symmetrical triangulation is generated in the next generation of simplices. This is done by an ordering of edges and using the cycle $(123 \ldots n-1)$ to ensure that we always split every simplex by a hyperplane that goes through a child vertex on the longest edge of simplex and every other vertex in the parent simplex that does not have incidence on the edge. Figure 4.5 demonstrates the symmetry of this sampling in $n = 2$ where the longest edge in the initial triangulation was sampled. Here an iteration is defined as any generation of sub-triangulations that provides a triangulation symmetrical to the initial triangulation. An implementation of this sampling sequence is available at Endres (2016–a).

In this publication we will use both the Sobol and the hypercube triangulation sampling sequences. Sobol provides a more direct comparison to the TGO algorithm while the second sequence is more similar to the DISIMPL-v algorithm. We will refer to the different uses of sampling sequences as SHGO-Sobol and SHGO-Simpl in the experimental results section in Section 5

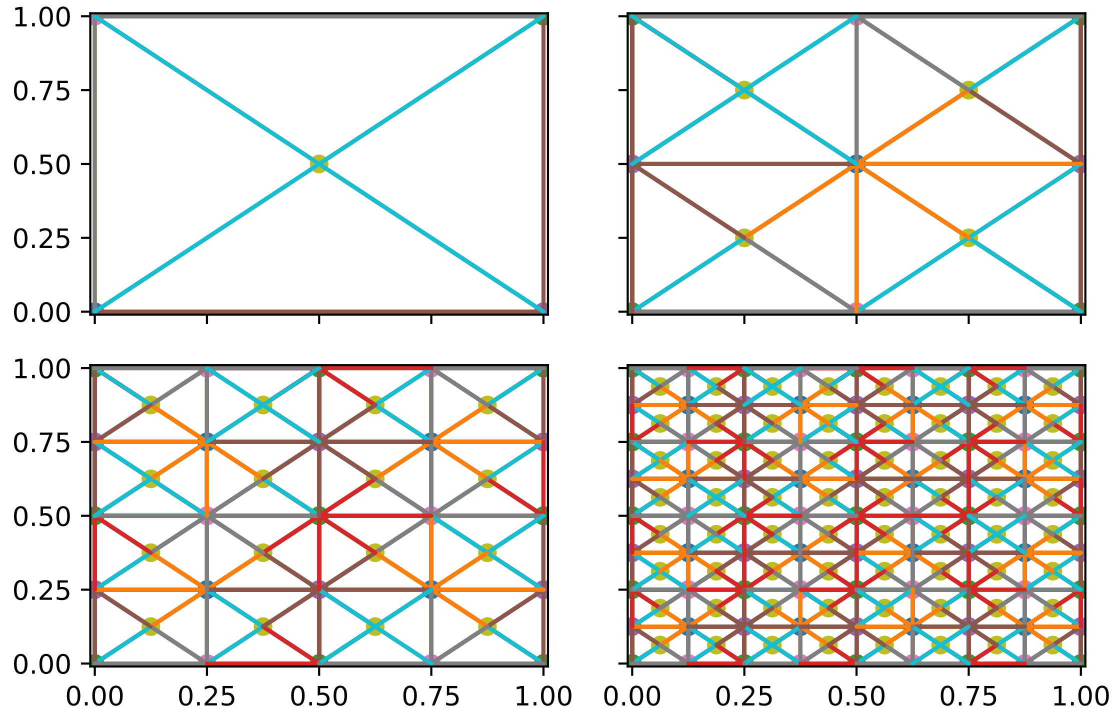

Figure 4.5: Triangulation of a unit hypercube shown in 2 dimensions for 4 iterations

4.6 Theoretical comparison to the DISIMPL algorithm

The DISIMPL algorithm developed by Paulavičius & Žilinskas Paulavičius & Žilinskas (2014b,a); Paulavičius et al. (2014) is based on spatial partitioning of the search space. DISIMPL-v in particular should have a similar initial complex as SHGO-Simpl for box problems since this algorithm samples on the vertices of the simplicial complex (while DISIMPL-c samples at the geometric centre of the simplices which is more appropriate for higher dimensional problems). The graph structure of DISIMPL-v can thus be used to construct the directed complex $\mathcal{H}$ and the homological properties can be calculated and applied. An example of one such application is given in the following paragraph.

At every iteration of the DISIMPL algorithm potentially optimal simplices are selected for refinement by considerations the Lipschitz properties of the optimisation problem. In general a combination of promising simplices with good function evaluations (related to local exploration of the search space) and simplices with larger hypervolumes (related to global exploration of the search space). Gb-DISIMPL Paulavičius et al. (2014) is a very promising acceleration technique accomplished by switching between a "global phase" and a "usual phase". The global phase is focused on exploring simplices with larger hyper volumes and excludes smaller simplices which are potentially optimal in the usual phase. This technique prevents excessive evaluations near local minima as demonstrated in Paulavičius et al. (2014). Local minima can put a "drag" on the progress of refining the minimum because the algorithm selects many neighbouring simplices that are slightly worse on the function values, but also slightly larger in volume. A meta-parameter is used in Gb-DISIMPL to select the simplices to be excluded in the global phase and was shown in Paulavičius et al. (2014) to be very efficient. However, using knowledge from the directed complex of $\mathcal{H}$, the domain containing these simplices near the local minima could also be identified more explicitly through a Sperner labelling if the function is known to be Lipschitz smooth.

4.7 Algorithm implementation

We consider two modes for the SHGO algorithm. In the first a finite number of sampling points N are specified and sampling is continued until an Ω set of cardinality N is produced and no further sampling occurs. This method is demonstrated by Algorithm 2. The main reason for this algorithm is to present a more direct comparison to TGO that can be used in numerical experiments.

For the purposes of global optimisation and local minima exploration Algorithm 3 is more appropriate. By continuously calculating the $\mathbf{H}_1(\mathcal{H})$ homology group several termination criteria can be used to end the sampling. For example if the amount of local minima is known the sampling can be terminated once $|\mathcal{M}|$ is large enough. Another example with many possible heuristics is tracking the historical difference in $|\mathcal{M}|$ over $|\mathcal{P}|$ and terminating sampling if $|\mathcal{M}|$ is unchanged after a certain increase in $|\mathcal{P}|$. In optimisation problems where the global minimum is known we can also use the stopping criteria such as the one defined by Paulavičius & Žilinskas (2016) .

$$pe = 100\% \times \begin{cases} \frac{\min\{\mathcal{F}\}-f^*}{|f^*|}, & f* \neq 0 \\ \min\{\mathcal{F}\}, & f^* = 0 \end{cases}$$

Here $\min\{\mathcal{F}\}$ is the minimum function evaluation obtained including values obtained in the output of the local minimisation step as shown in the algorithm. Whatever termination criterion is used it requires an input $\mathbf{H}_1(\mathcal{H})$ or $\min\{\mathcal{F}\}$ and should output a Boolean, we will refer to this function as $\mathbf{TERM}(\mathbf{H}_1(\mathcal{H}), \min\{\mathcal{F}\})$ in Algorithm 3. In the practical implementation of the algorithm the user can also specify a finite number of iterations and/or sampling points. This functionality has been programmed into the $\mathbf{TERM}(\mathbf{H}_1(\mathcal{H}), \min\{\mathcal{F}\})$ function.

Open source python implementations of both of these algorithms are available and were published under a MIT compatible license Endres (2016–a).

Algorithm 2 SHGO finite sampling algorithm

```
1: procedure Initialisation
2:     Input an objective function $f$, constraint functions $\mathbf{g}$ and variable bounds and $[\mathbf{l}, \mathbf{u}]^n$.
3:     Input $N$ initial sampling points.
4:     Define a sampling sequence that generates a set $\mathcal{X}$ of sampling points in the unit hypercube space $[\mathbf{0}, \mathbf{1}]^n$
5: end procedure
6: procedure Initial sampling
7:     $\mathcal{P} = \emptyset$
8:     while $|\mathcal{P}| < N$ do
9:         Generate $N - |\mathcal{P}|$ sequential sampling points $\mathcal{X} \subset \mathbb{R}^n$
10:        Stretch $\mathcal{X}$ over the lower and upper bounds $[\mathbf{l}, \mathbf{u}]^n$
11:        $\mathcal{P} = \{\mathcal{X}_i \mid \mathbf{g}(\mathcal{X}_i) \geq 0, \forall \mathcal{X}_i \in \mathcal{X}\} \cup \mathcal{P}$    ▷ (Find $\mathcal{P}$ in the feasible subset $\Omega$ by discarding any points mapped outside the linear constraints $g$ and adding to the current set of $\mathcal{P}$.)
12:        Set $\mathcal{X} = \emptyset$
13:    end while
14:    Find $\mathcal{F}$ from the objective function $f : \mathcal{P} \rightarrow \mathcal{F}$
15: end procedure
16: procedure Construct directed complex $\mathcal{H}$
17:    Calculate $\mathcal{H}$ from $h : \mathcal{P} \rightarrow \mathcal{H}$
18: end procedure
19: procedure Construct $\mathcal{M}$
20:    Find $\mathcal{M}$ from Definition 20.
21: end procedure
22: procedure Local minimisation
23:    Calculate the approximate local minima of $f$ using a local minimisation routine with the elements of $\mathcal{M}$ as starting points.    ▷ These local minimisations can be performed in parallel.
24: end procedure
25: procedure Process return objects
26:    Order the final outputs of the minima of $f$ found in the local minimisation step to find the approximate global minimum.
27: end procedure
28:
29: return the approximate global minimum and a list of all the minima found in the local minimisation step.
```

Algorithm 3 SHGO homology group growth algorithm

```
 1: procedure Initialisation
 2:     Input an objective function $f$, constraint functions $\mathbf{g}$ and variable bounds and $[\mathbf{l}, \mathbf{u}]^n$.
 3:     Input $N$ initial sampling points.
 4:     Define a sampling sequence that generates a set $\mathcal{X}$ of sampling points in the unit hypercube space $[\mathbf{0}, \mathbf{1}]^n$
 5:     Define the empty set $\mathcal{M}^E = \emptyset$ of vertices evaluated by a local minimisation.
 6: end procedure
 7: while $\mathbf{TERM}(\mathbf{H}_1(\mathcal{H}), \min\{\mathcal{F}\})$ is False do
 8:     procedure Sampling
 9:         $\mathcal{P} = \emptyset$
10:         while $|\mathcal{P}| < N$ do
11:             Generate $N - |\mathcal{P}|$ sequential sampling points $\mathcal{X} \subset \mathbb{R}^n$
12:             Stretch $\mathcal{X}$ over the lower and upper bounds $[\mathbf{l}, \mathbf{u}]^n$
13:             $\mathcal{P} = \{\mathcal{X}_i \mid \mathbf{g}(\mathcal{X}_i) \geq 0, \forall \mathcal{X}_i \in \mathcal{X}\} \cup \mathcal{P}$    ▷ (Find $\mathcal{P}$ in the feasible subset $\Omega$ by discarding any points mapped outside the linear constraints $g$ and adding to the current set of $\mathcal{P}$.)
14:             Set $\mathcal{X} = \emptyset$
15:         end while
16:         Find $\mathcal{F}$ from the objective function $f : \mathcal{P} \to \mathcal{F}$ for any new points in $\mathcal{P}$
17:     end procedure
18:     procedure Construct/append directed complex $\mathcal{H}$
19:         Calculate $\mathcal{H}$ from $h : \mathcal{P} \to \mathcal{H}$    ▷ (If $\mathcal{H}$ was already constructed new points in $\mathcal{P}$ are incorporated into the triangulation.)
20:         Calculate $\mathbf{H}_1(\mathcal{H})$
21:     end procedure
22:     procedure Construct $\mathcal{M}$
23:         Find $\mathcal{M}$ from Definition 20.
24:     end procedure
25:     procedure Local minimisation
26:         Calculate the approximate local minima of $f$ using a local minimisation routine with the elements of $\mathcal{M} \setminus \mathcal{M}^E$ as starting points.    ▷ Process the most promising points first.
27:         $\mathcal{M}^E = \mathcal{M}^E \cap \mathcal{M}$    ▷ This excludes the evaluation any element $v_i \in \mathcal{M}$ that is known to be the only point that in the domain $\partial \mathrm{st}(v_j)$ where $v_j$ is known to any point already used as a starting point in Step 27. If any new $v_i \in \mathcal{M}$ not in $\mathcal{M}^E$ is known to be the only point $\partial \mathrm{st}(v_j)$ it can also be excluded.
28:         Add the function outputs of the local minimisation routine to $\mathcal{F}$
29:     end procedure
30:     Find new value of $\mathbf{TERM}(\mathbf{H}_1)(\mathcal{H}, \min\{\mathcal{F}\})$
31: end while
32: procedure Process return objects
33:     Order the final outputs of the minima of $f$ found in the local minimisation step to find the approximate global minimum.
34: end procedure
35:
36: return the approximate global minimum and a list of all the minima found in the local minimisation step.
```

CHAPTER 5

Experimental Results

5.1 Comparison to with linear constraints

In this section we provide experimental comparisons on 22 linearly constrained problems comparing the SHGO, TGO, Lc-DISIMPL Paulavičius & Žilinskas (2016), PSwarm Vaz & Vicente (2009) and DIRECT-L1 Finkel (2003) algorithms. Note that the data for the Lc-DISIMPL, PSwarm and DIRECT-L1 algorithms was taken from Paulavičius & Žilinskas (2016). The same percentage error of $pe = 0.01\%$ used by Paulavičius & Žilinskas (2016) was also used in this publication. To provide a fair comparison of TGO to SHGO and the other solvers the TGO algorithm was modified to stop sampling when it produced a minimiser that lead to the global minimum of the problem. Table 5.1 shows the results. Here f.e. is the total number of objective function evaluations required to solve the function and p.f.e. is the total number of penalty function evaluations. Paulavičius & Žilinskas (2016) used DIRECT-L1 with the 3 different penalty parameters (p.p.) shown in the table. The PSwarm solver was run 10 times for each test problem.

The SHGO-Simpl, SHGO-Sobol and TGO (using Henderson's formula for k_c) algorithms were able to solve all 22 problems. The lowest average number of function evaluations was achieved by SHGO-Simpl followed by SHGO-Sobol and TGO. It can be observed that Lc-DISIMPL-v achieved a better performance than any other algorithm for the horst-1 to horst-6, hs024, hs035, s232, s250 and bunnag2 problems. As noted in Paulavičius & Žilinskas (2016) the initial triangulation of Lc-DISIMPL-v evaluates the function values at the vertices of the simplices and therefore for some of the tested problems the solutions were found after initial triangulation on one of the vertices of the feasible region. It is also possible to initiate SHGO with such an initial triangulation by definition the first few vertices in $\mathcal{X}$ as the intercepts of the linear constraints in a similar way to Paulavičius & Žilinskas (2016) and then continuing to add sampling points as normal.

Table 5.2 provides additional information for SHGO and TGO including the total number of function evaluations required by the algorithm to solve the problem (f.e.), the number of minimisers generated as starting points by the algorithm (nlmin), the number of unique local minima mapped by the algorithm (nulmin) and the total processing time (runtime) in seconds.

It can be seen that neither of the SHGO algorithms produced more starting points leading to the same local minima as predicted by the theory for adequately sampled function surfaces. On the contrary TGO produced more than one starting point in the same locally convex domain on some test problems which lead to extra function evaluations, producing a poorer overall performance. While SHGO-Simpl had the lowest number of average function evaluations, a higher processing run time is observed compared to the other 2 algorithms. This can be explained by the fact the triangulation code for the sampling has not yet been optimised which consumed most of the run time. SHGO-Sobol and TGO use the same sampling generation code and it is observed that SHGO-Sobol has a lower processing run time as expected.

The source code used to produce these results including the scripts that run the test benchmarking suite is publically available at Endres (2016–a). The specifications of the system used to run the test problems can be found in Appendix A.

Table 5.1: Function evaluation comparisons for test problems with linear constraints.

	shgo-		tgo	Lc-DSIMPL-[c]		PSwarm[c]						DIRECT-L1[c]		
	-simpl	-sobol		-v	-c	Minimum		Average		Maximum		p.p. = 10	p.p. = 10^2	p.p. = 10^6
Problem	f.e.	f.e.	f.e.	f.e.	f.e.	f.e.	p.f.e	f.e.	p.f.e.	f.e.	p.f.e	f.e	f.e.	f.e.
horst-1	97	24	34	7	249	167	182	$1329^{b(3)}$	$1343^{b(3)}$	$4100^{b(3)}$	$4101^{b(3)}$	287^a	3689	>100000
horst-2	10	11	11	5	171	160	176	424	492	768	867	265^a	10829	>100000
horst-3	6	7	6	5	249	42	43	44	45	46	47	5^a	591	617
horst-4	10	25	24	8	260	90	179	114	194	129	211	58293^a	>100000	>100000
horst-5	20	15	15	8	259	106	150	134	192	214	302	7^a	>100000	>100000
horst-6	22	59	77	10	284	90	172	110	192	133	227	11^a	739^a	>100000
horst-7	10	15	13	10	220	188	201	380	403	919	957	7^a	71^a	>100000
hs021	24	23	23	189	133	110	110	189	192	392	405	97	97	97
hs024	24	15	36	3	141	101	153	118	172	138	195	19^a	57^a	>100000
hs035	37	41	35	630	721	266	311	316	369	327	373	>100000	>100000	>100000
hs036	105	20	103	8	314	179	179	396	401	561	574	25^a	49^a	>100000
hs037	72	63	258	186	9129	127	131	160	167	201	574	7^a	7^a	>100000
hs038	225	1029	389	3379	>100000	53662	54445	58576	59821	65677	67660	7401	5885	6511
hs044	199	35	51	20	440	$148^{b(9)}$	$218^{b(9)}$	$186^{b(9)}$	$281^{b(9)}$	$201^{b(9)}$	$299^{b(9)}$	90283	>100000	>100000
hs076	56	37	44	548	4794	132	198	203	286	275	341	19135	>100000	>100000
s224	166	165	165	49	463	105	107	121	122	157	158	7^a	431	457
s231	99	99	383	2137	655	542	1011	2366	3020	4116	4800	1261	1209	43341
s232	24	15	22	3	141	105	144	119	171	162	236	19^a	57^a	>100000
s250	105	20	103	8	314	296	296	367	375	495	498	25^a	49^a	>100000
s251	72	63	258	186	9127	83	84	129	137	175	180	7^a	7^a	>100000
bunnag1	34	47	39	630	721	132	142	214	228	411	438	1529	1495	1463
bunnag2	46	36	35	16	500	150	153	252	259	410	426	>100000	>100000	>100000
Average	66	88	100	366	>5877	2590	2672	3011	3130	3637	3812	>17213	>28421	>75113

a result is outside the feasible region
b(*t*) *t* out of 10 times the global solution was not reached
c results produced by Paulavičius & Žilinskas (2016)

Table 5.2: Total and average performance over all 22 test problems.

problem	name	f.e.	nlmin	nulmin	runtime (s)
All	shgo-simpl	1463	26	26	0.27294
	shgo-sobol	1864	23	23	0.11225
	tgo	2123	29	25	0.093607
Average	shgo-simplicial	65	1	1	0.012852
	shgo-sobol	88	1	1	0.004144
	tgo	100	1	1	0.004542

5.2 Function evaluations and comparison to other open source global optimisation algorithms

In this section we present numerical experiments comparing the SHGO and TGO algorithms with the SciPy implementations Jones et al. (2001–) of basinhopping (BH) Li and Scheraga (1987); Wales (2003); Wales and Doye (1997); Wales and Scheraga (1999) and differential evolution (DE) Storn and Price (1997). These algorithms were chosen both because the open source versions are readily available in the SciPy project and because BH is commonly used in energy surface optimisations Wales (2015) from which the motivation for developing SHGO grew. DE has also been applied in optimising Gibbs energy surfaces for phase equilibria calculations Zhang & Rangaiah (2011). The optimisation problems in Appendix A were selected from the SciPy global optimisation benchmarking test suite (Adorio and Dilman, 2005; Gavana, 2016; Jamil and Yang, 2013; Mishra, 2007, 2006; NIST, 2016). The test suite contains multi-modal problems with box constraints, they are described in detail in Gavana (2016). We again used the stopping criteria $pe = 0.01\%$ for SHGO and TGO. For the stochastic algorithms (BH and DE) the starting points provided by the test suite were used. For every test the algorithm was terminated if the global minimum was not found after 10 minutes of processing time and the test was flagged as a fail. For comparisons we used normalised performance profiles Dolan and Moré (2002) using function evaluations and processing time as performance criteria. In total 180 test problems were used.

From Fig. 5.1 it can be observed that for this problem set SHGO-Sobol was the best performing algorithm, followed closely by TGO and SHGO-Simpl. Fig. 5.2 provides a clearer comparison between these three algorithms. While the performance of all 3 algorithms are comparable, SHGO-Sobol tends to outperform TGO, solving more problems for a given number of function evaluations. This is expected since, for the same sampling point sequence, TGO produced more than one starting point in the same locally convex

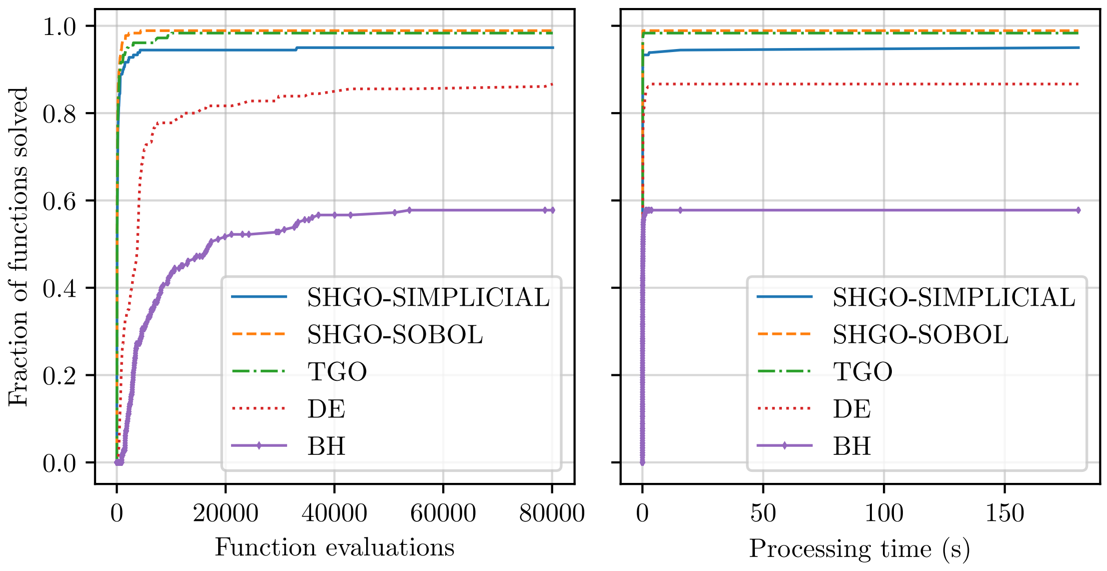

Figure 5.1: Performance profiles for SHGO, TGO, DE and BH on SciPy benchmarking test suite

domain on some test problems which leads to extra function evaluations. In total TGO produced 403 minima of which only 393 minima were unique while all of the 225 minima produced by SHGO-Sobol were unique. SHGO-Simpl produced 238 of which all 238 were unique. It is apparent that SHGO-Simpl performed worse compared to the other sampling methods despite a better performance on the test problem set with linear constraints. There are two reasons for this result. First of all the uniformity properties of the Sobol sequence hold only for hypercubes, therefore it is lost for geometries defined by the search spaces inside linear constraints. Secondly the current code for the triangulation of the simplex cannot add only one sampling point per iteration, but must split all the simplices until the symmetry of the entire complex is restored. This leads to a much higher number of function evaluations during the sampling step of the algorithm.

The Table in Appendix A shows the raw numerical results. Note that, unlike the data in performance profiles, failed test runs did not get set to the worst case performance criteria by any solver (in order to preserve the raw data). Therefore the total and average function evaluations and processing times are misleading. The Table is mostly useful for comparisons on a particular test problem as well as comparing the total number of minima and unique minima found.

5.3 Invariance and optimum minimiser pool

The following 4 optimisation test problems were used to demonstrate the applications of Theorem 5 and to show the minimiser pool growth compared to TGO over a large number of sampling points. The results plotted in Figure 5.3 shows that SHGO performed as

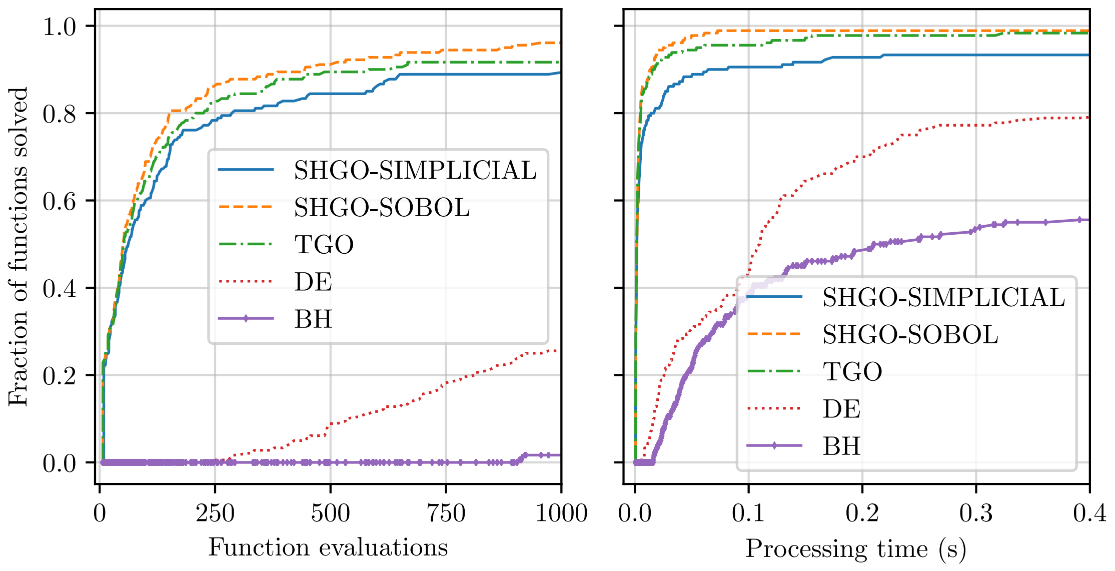

Figure 5.2: Performance profiles zoomed in to the range of $f.e. = [0, 1000]$ function evaluations and $[0, 0.4]$ seconds run time

expected with the minimiser pool staying at the optimum cardinality to map all the local minima once the sampling is adequate as well as the shortcomings of the TGO especially in the higher dimensional test problems where the the minimiser pool tends to grow rapidly with the number sampling points N.

The Ursem01 function for two dimensions is defined as follows Gavana (2016)

$$f(\mathbf{x}) = -\sin(2x_1 - 0.5\pi) - 3\cos(x_2) - 0.5x_1, \ \mathbf{x} \in \Omega = [0, 9] \times [-2, 2] \tag{5.1}$$

The Paraboloid function for six dimensions is defined as follows

$$f(\mathbf{x}) = \sum_{i=1}^{6} x_i^2, \ \mathbf{x} \in \Omega = [-10, 10]^6 \tag{5.2}$$

The Bird function for two dimensions is defined as follows Gavana (2016)

$$f(\mathbf{x}) = (x_1 - x_2)^2 + e^{[1-\sin(x_1)]^2}\cos(x_2) + e^{[1-\cos(x_2)]^2}\sin(x_1), \\ \mathbf{x} \in \Omega = [-2\pi, 2\pi]^2 \tag{5.3}$$

The Schwefel01 function for six dimensions is defined as follows Gavana (2016)

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} x_i^2\right)^{\sqrt{\pi}}, \ \mathbf{x} \in \Omega = [-100, 100]^6 \tag{5.4}$$

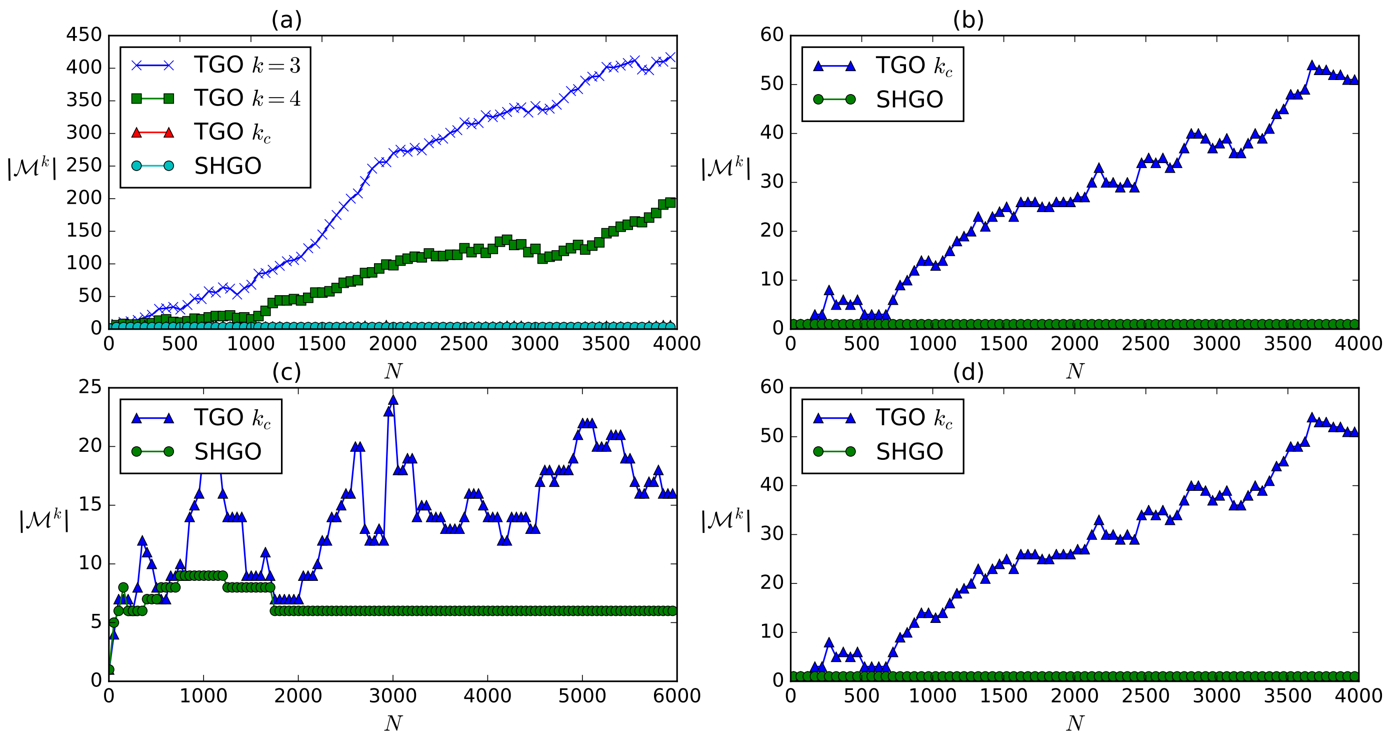

Figure 5.3: (a) The minimiser pool growth of the TGO and SHGO algorithms for the smooth objective function described in Example 3 and restated in Equation (5.1) for convenience, the SHGO never increases above the optimum of $|\mathcal{M}| = 3$, for TGO 3 different values of the k parameter are shown. (b) The minimiser pool growth for the six dimensional Paraboloid problem defined by Equation (5.2), note that even though the problem has only one minimum, the minimiser pool for TGO set at $k = k_c$ tends to increase for increasing sampling points N. In general this problem is exacerbated in higher dimensions while SHGO stays at the optimum $|\mathcal{M}| = 1$. The TGO minimiser pool for $k = 3$ and $k = 4$ are not shown here because the minimiser pool grows too rapidly. (c) The minimiser pool growth for the two dimensional Bird problem defined by Equation (5.3), an important observation here is that $|\mathcal{M}|$ is higher than optimum for SHGO before the sampling is adequate as defined by Equation (5) which happens at the after there are $N = 1722$ Sobol sequenced points after which $|\mathcal{M}|$ stays at the optimum value equal to the number of unique local minima with increasing N. (d) The minimiser pool growth for the six dimensional Schwefel01 problem defined by Equation (5.4), here again $|\mathcal{M}|$ for TGO set at k_c grows rapidly with N while $|\mathcal{M}|$ for SHGO stays constant at the optimum.

CHAPTER 6

Concluding remarks

The SHGO algorithm developed here shows promising properties and performance. On problems with linear constraints it was shown to provide competitive results to the TGO, Lc-DISIMPL, PSwarm and DIRECT-L1 algorithms. The use of a simplicial complex provides access to a wealth of tools from combinatorial topology and the growing field of computational homology. We are hopeful that these will drive further extensions and development of the algorithm. Many challenges remain such as finding the most appropriate sampling sequences for different classes of problems and finding computer resource efficient triangulation schemes. Due to the useful characterisations of objective function hypersurfaces provided by the homology groups of the simplicial complex SHGO allows an optimisation practitioner with a useful visual tool for understanding and efficiently solving higher dimensional black and grey box optimisation problems.

The main initial driving force behind the development of this algorithm grew out of a need for efficient, deterministic and reliable global optimisation methods for applications in phase equilibria modelling and calculations. However, the SHGO algorithm described here is appropriate for solving a wider class of global optimisation problems both those where mapping all the local minima is of interest and where only the global optimum is needed. It is especially appropriate for computationally expensive black and grey box functions common in science and engineering as described for example by Shan and Wang (2010).

Some key features of SHGO are that when the optimisation search space is adequately sampled and enough information is available to determine that all local minima have been mapped it is guaranteed that only one starting point for every locally convex domain will be produced by the algorithm. Note that in optimisation problems where the number of local minima is known, the sampling can stop and the local minimisation step started without superfluous function evaluations while for optimisation problems with an unknown number of local minima is unknown (and thus we can never truly know if all local minima has been found for any finite number of sampling) the guarantee still holds that that SHGO will not produce superfluous starting points that lead to the same stationary points. In addition because the homology groups can be calculated as sampling progresses an optimisation practitioner can both visualise the extent of the optimisation problem's multi-modality and use intelligent stopping criteria for the sampling stage.

APPENDIX A

Numerical results for selected optimisation problems

Table A.1 shows respectively: the name of the optimisation test problem (Problem), the name of the algorithm (Alg), number of dimensions (n) of the optimisation problem, the number of function evaluations required by the algorithm to solve the problem (nfev), the number of minimisers generated as starting points by the algorithm (nlmin), the number of unique local minima mapped by the algorithm (nulmin), whether successful convergence to the global minima was achieved (Success), the CPU run time measured in seconds (Runtime) and finallythe number of function evaluations per unique local minima (nfev/nulmin). For all these test problems the algorithm was terminated if the algorithm ran for longer than 10 minutes.

The optimisation runs were done on a computer with the following specifications:

- CPU: Intel Core i7-6700K CPU @ 4.2GHz
- Kernel: x86_64 Linux 4.12.10-1-ARCH
- RAM: 15973MiB

Table A.1: Comparison of the performance of SHGO, TGO, BH and DE over a wide selection of optimisation test problems. The columns are the name of the optimisation test problem, the algorithm (Alg) used, the number of dimensions (n) of the optimisation problem, the number of function evaluations (nfev), the number of minimisers generated (nlmin), the number of unique local minima (nulmin), successful convergence to the global minima and the total CPU run time (runtime).

All	bh	0	1358408	0	0	NaN	NaN
	de	0	934804	0	0	NaN	NaN
	shgo-simplicial	0	72240	238	238	NaN	NaN

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	shgo-sobol	0	29694	225	225	NaN	NaN
	tgo	0	63533	403	393	NaN	NaN
Average	bh	0	7546	0	0	NaN	0.108971
	de	0	5193	0	0	NaN	0.188172
	shgo-simplicial	0	401	1	1	NaN	1.115545
	shgo-sobol	0	164	1	1	NaN	0.004778
	tgo	0	352	2	2	NaN	0.008672
Ackley01	bh	2	16107	0	0	True	0.298839
	de	2	3423	0	0	True	0.190420
	shgo-simplicial	2	54	1	1	True	0.001750
	shgo-sobol	2	52	1	1	True	0.041898
	tgo	2	52	1	1	True	0.001998
Ackley02	bh	2	11844	0	0	True	0.090117
	de	2	456	0	0	True	0.010810
	shgo-simplicial	2	90	1	1	True	0.001905
	shgo-sobol	2	88	1	1	True	0.001738
	tgo	2	88	1	1	True	0.001615
Ackley03	bh	2	2370	0	0	False	0.040504
	de	2	421	0	0	True	0.013166
	shgo-simplicial	2	59	1	1	True	0.001444
	shgo-sobol	2	57	1	1	True	0.001529
	tgo	2	57	1	1	True	0.001432
Adjiman	bh	2	2070	0	0	False	0.046875
	de	2	532	0	0	True	0.037358
	shgo-simplicial	2	26	1	1	True	0.003626
	shgo-sobol	2	36	1	1	True	0.004907
	tgo	2	36	1	1	True	0.004410
Alpine01	bh	2	32928	0	0	True	0.303166
	de	2	4423	0	0	True	0.138957
	shgo-simplicial	2	55	1	1	True	0.001360
	shgo-sobol	2	53	1	1	True	0.001466
	tgo	2	53	1	1	True	0.001400
Alpine02	bh	2	1617	0	0	True	0.024743

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	de	2	492	0	0	True	0.014723
	shgo-simplicial	2	153	5	5	True	0.005421
	shgo-sobol	2	62	1	1	True	0.001830
	tgo	2	108	3	3	True	0.002290
BartelsConn	bh	2	19857	0	0	True	0.205387
	de	2	1282	0	0	True	0.036180
	shgo-simplicial	2	55	1	1	True	0.001298
	shgo-sobol	2	53	1	1	True	0.001491
	tgo	2	53	1	1	True	0.001306
Beale	bh	2	6306	0	0	False	0.045135
	de	2	4803	0	0	True	0.127161
	shgo-simplicial	2	63	1	1	True	0.001226
	shgo-sobol	2	61	1	1	True	0.001339
	tgo	2	61	1	1	True	0.001239
BiggsExp02	bh	2	3009	0	0	True	0.079360
	de	2	4003	0	0	True	0.177575
	shgo-simplicial	2	147	2	2	True	0.005318
	shgo-sobol	2	128	1	1	True	0.004324
	tgo	2	128	1	1	True	0.004133
BiggsExp03	bh	3	5812	0	0	True	0.134723
	de	3	10564	0	0	True	0.492391
	shgo-simplicial	3	145	1	1	True	0.007089
	shgo-sobol	3	151	1	1	True	0.005064
	tgo	3	151	1	1	True	0.004900
BiggsExp04	bh	4	13095	0	0	True	0.295152
	de	4	29765	0	0	True	1.368383
	shgo-simplicial	4	1091	1	1	True	0.167113
	shgo-sobol	4	384	1	1	True	0.011662
	tgo	4	384	1	1	True	0.011372
BiggsExp05	bh	5	14346	0	0	False	0.468451
	de	5	7632	0	0	False	0.461430
	shgo-simplicial	5	607	1	1	True	0.023766
	shgo-sobol	5	583	1	1	True	0.019824
	tgo	5	583	1	1	True	0.019717

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
Bird	bh	2	2421	0	0	False	0.038661
	de	2	695	0	0	True	0.021481
	shgo-simplicial	2	42	1	1	True	0.001750
	shgo-sobol	2	43	1	1	True	0.001341
	tgo	2	43	1	1	True	0.001209
Bohachevsky1	bh	2	3510	0	0	True	0.037407
	de	2	2763	0	0	True	0.077462
	shgo-simplicial	2	9	1	1	True	0.000461
	shgo-sobol	2	7	1	1	True	0.000564
	tgo	2	7	1	1	True	0.000511
Bohachevsky2	bh	2	3471	0	0	True	0.037333
	de	2	2923	0	0	True	0.080887
	shgo-simplicial	2	9	1	1	True	0.000520
	shgo-sobol	2	7	1	1	True	0.000620
	tgo	2	7	1	1	True	0.000496
Bohachevsky3	bh	2	3438	0	0	True	0.033962
	de	2	3043	0	0	True	0.081093
	shgo-simplicial	2	9	1	1	True	0.000478
	shgo-sobol	2	7	1	1	True	0.000603
	tgo	2	7	1	1	True	0.000481
BoxBetts	bh	3	8096	0	0	True	0.181494
	de	3	11944	0	0	True	0.525109
	shgo-simplicial	3	89	1	1	True	0.003011
	shgo-sobol	3	76	1	1	True	0.002723
	tgo	3	76	1	1	True	0.002527
Branin01	bh	2	2229	0	0	True	0.027123
	de	2	615	0	0	True	0.017054
	shgo-simplicial	2	39	1	1	True	0.000982
	shgo-sobol	2	37	1	1	True	0.001067
	tgo	2	37	1	1	True	0.000935
Branin02	bh	2	2094	0	0	True	0.031920
	de	2	735	0	0	True	0.022043
	shgo-simplicial	2	111	2	2	True	0.004283
	shgo-sobol	2	44	1	1	True	0.001358

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	tgo	2	71	2	2	True	0.001664
Brent	bh	2	915	0	0	True	0.016313
	de	2	6443	0	0	True	0.176477
	shgo-simplicial	2	9	1	1	True	0.000487
	shgo-sobol	2	7	1	1	True	0.000596
	tgo	2	7	1	1	True	0.000502
Brown	bh	2	1857	0	0	True	0.038897
	de	2	4083	0	0	True	0.146456
	shgo-simplicial	2	34	1	1	True	0.001163
	shgo-sobol	2	36	1	1	True	0.001331
	tgo	2	36	1	1	True	0.001246
Bukin02	bh	2	663	0	0	False	0.014341
	de	2	815	0	0	True	0.021064
	shgo-simplicial	2	20	1	1	True	0.000664
	shgo-sobol	2	18	1	1	True	0.000764
	tgo	2	18	1	1	True	0.000643
Bukin04	bh	2	17166	0	0	True	0.098578
	de	2	4103	0	0	True	0.110858
	shgo-simplicial	2	26	1	1	True	0.000751
	shgo-sobol	2	24	1	1	True	0.001064
	tgo	2	24	1	1	True	0.000798
Bukin06	bh	2	22014	0	0	False	0.179138
	de	2	2623	0	0	False	0.075753
	shgo-simplicial	2	1007	5	5	True	0.021791
	shgo-sobol	2	741	3	3	True	0.012376
	tgo	2	1169	5	5	True	0.017350
CarromTable	bh	2	1899	0	0	False	0.035184
	de	2	972	0	0	True	0.034849
	shgo-simplicial	2	36	1	1	True	0.001454
	shgo-sobol	2	31	1	1	True	0.001087
	tgo	2	31	1	1	True	0.000984
Cigar	bh	2	8193	0	0	True	0.088217
	de	2	3743	0	0	True	0.124895
	shgo-simplicial	2	20	1	1	True	0.000687

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	shgo-sobol	2	18	1	1	True	0.000797
	tgo	2	18	1	1	True	0.000693
Colville	bh	4	12965	0	0	True	0.103124
	de	4	29685	0	0	True	0.837259
	shgo-simplicial	4	225	1	1	True	0.014057
	shgo-sobol	4	1029	2	2	True	0.060954
	tgo	4	3039	10	2	True	0.054228
Corana	bh	4	2555	0	0	False	0.073994
	de	4	4085	0	0	True	0.192973
	shgo-simplicial	4	23	1	1	True	0.002139
	shgo-sobol	4	12	1	1	True	0.000986
	tgo	4	12	1	1	True	0.000847
CosineMixture	bh	2	348	0	0	False	0.014602
	de	2	1166	0	0	True	0.037936
	shgo-simplicial	2	17	1	1	True	0.001120
	shgo-sobol	2	7	1	1	True	0.000641
	tgo	2	7	1	1	True	0.000552
CrossInTray	bh	2	1578	0	0	False	0.028986
	de	2	489	0	0	True	0.019297
	shgo-simplicial	2	69	1	1	True	0.003238
	shgo-sobol	2	33	1	1	True	0.001108
	tgo	2	33	1	1	True	0.001020
CrossLegTable	bh	2	17355	0	0	True	0.209052
	de	2	4783	0	0	False	0.154005
	shgo-simplicial	2	9	1	1	True	0.000545
	shgo-sobol	2	7	1	1	True	0.000651
	tgo	2	7	1	1	True	0.000640
CrownedCross	bh	2	17130	0	0	False	0.210953
	de	2	4263	0	0	False	0.136233
	shgo-simplicial	2	9	1	1	True	0.000558
	shgo-sobol	2	7	1	1	True	0.000644
	tgo	2	7	1	1	True	0.000520
Cube	bh	2	8529	0	0	True	0.053030
	de	2	5243	0	0	True	0.125904

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	shgo-simplicial	2	146	1	1	True	0.002060
	shgo-sobol	2	144	1	1	True	0.002167
	tgo	2	144	1	1	True	0.002071
Damavandi	bh	2	1566	0	0	False	0.026171
	de	2	535	0	0	False	0.015558
	shgo-simplicial	2	578	2	2	True	0.034168
	shgo-sobol	2	60	2	2	True	0.001825
	tgo	2	97	2	2	True	0.002025
DeVilliersGlasser01	bh	4	16805	0	0	True	0.554629
	de	4	24265	0	0	True	1.200274
	shgo-simplicial	4	439	2	2	True	0.024691
	shgo-sobol	4	446	1	1	True	0.044395
	tgo	4	667	9	9	True	0.023041
Deb01	bh	2	5565	0	0	True	0.065591
	de	2	1532	0	0	True	0.048103
	shgo-simplicial	2	28	1	1	True	0.001334
	shgo-sobol	2	18	1	1	True	0.000828
	tgo	2	18	1	1	True	0.000741
Deb03	bh	2	5310	0	0	False	0.076148
	de	2	40103	0	0	False	1.453880
	shgo-simplicial	2	4234	4	4	True	0.082374
	shgo-sobol	2	83	1	1	True	0.002579
	tgo	2	1476	2	2	True	0.029756
Decanomial	bh	2	9465	0	0	True	0.117149
	de	2	3383	0	0	True	0.107160
	shgo-simplicial	2	256	1	1	True	0.005111
	shgo-sobol	2	200	1	1	True	0.004302
	tgo	2	200	1	1	True	0.004083
Deceptive	bh	2	441	0	0	False	0.017110
	de	2	913	0	0	True	0.025888
	shgo-simplicial	2	449	9	9	True	0.017866
	shgo-sobol	2	533	4	4	True	0.012727
	tgo	2	667	5	5	True	0.015315
DeckkersAarts	bh	2	2844	0	0	True	0.026594

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	de	2	670	0	0	True	0.016102
	shgo-simplicial	2	86	1	1	True	0.003099
	shgo-sobol	2	99	2	2	True	0.002026
	tgo	2	167	2	2	True	0.002544
DefCorrSpring	bh	2	2055	0	0	True	0.043273
	de	2	892	0	0	True	0.032436
	shgo-simplicial	2	9	1	1	True	0.000695
	shgo-sobol	2	7	1	1	True	0.000676
	tgo	2	7	1	1	True	0.000560
DixonPrice	bh	2	3639	0	0	True	0.059777
	de	2	4563	0	0	True	0.160772
	shgo-simplicial	2	627	3	3	True	0.039069
	shgo-sobol	2	93	2	2	True	0.002869
	tgo	2	92	2	2	True	0.002412
Dolan	bh	5	51084	0	0	True	0.390904
	de	5	78692	0	0	True	2.479181
	shgo-simplicial	5	264	1	1	True	0.007559
	shgo-sobol	5	240	1	1	True	0.004052
	tgo	5	240	1	1	True	0.003832
DropWave	bh	2	2337	0	0	True	0.036634
	de	2	1012	0	0	True	0.032547
	shgo-simplicial	2	9	1	1	True	0.000526
	shgo-sobol	2	7	1	1	True	0.000634
	tgo	2	7	1	1	True	0.000524
Easom	bh	2	303	0	0	False	0.013139
	de	2	83	0	0	False	0.001927
	shgo-simplicial	2	2126	1	1	True	0.139017
	shgo-sobol	2	2210	1	1	True	0.049775
	tgo	2	2210	1	1	True	0.323876
EggCrate	bh	2	1935	0	0	False	0.025269
	de	2	3963	0	0	True	0.110584
	shgo-simplicial	2	9	1	1	True	0.000492
	shgo-sobol	2	7	1	1	True	0.000619
	tgo	2	7	1	1	True	0.000511

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
EggHolder	bh	2	1983	0	0	False	0.046486
	de	2	941	0	0	False	0.036749
	shgo-simplicial	2	616	2	2	True	0.042003
	shgo-sobol	2	233	4	4	True	0.007344
	tgo	2	293	5	5	True	0.008198
EAVD	bh	2	3219	0	0	True	0.031350
	de	2	1301	0	0	True	0.034577
	shgo-simplicial	2	33094	2	2	True	2.799852
	shgo-sobol	2	277	3	3	True	0.005578
	tgo	2	351	4	4	True	0.005266
Exp2	bh	2	2892	0	0	True	0.078883
	de	2	4003	0	0	True	0.184855
	shgo-simplicial	2	56	1	1	True	0.002597
	shgo-sobol	2	137	1	1	True	0.004752
	tgo	2	137	1	1	True	0.004762
Exponential	bh	2	1515	0	0	True	0.026072
	de	2	286	0	0	True	0.008401
	shgo-simplicial	2	9	1	1	True	0.000546
	shgo-sobol	2	7	1	1	True	0.000624
	tgo	2	7	1	1	True	0.000538
FreudensteinRoth	bh	2	5262	0	0	True	0.041467
	de	2	4103	0	0	True	0.100865
	shgo-simplicial	2	356	7	7	True	0.012095
	shgo-sobol	2	49	1	1	True	0.001195
	tgo	2	49	1	1	True	0.001008
Gear	bh	4	505	0	0	False	0.013874
	de	4	11445	0	0	True	0.336399
	shgo-simplicial	4	23	1	1	True	0.001465
	shgo-sobol	4	31	1	1	True	0.001925
	tgo	4	37	2	2	True	0.001130
Giunta	bh	2	2301	0	0	True	0.046305
	de	2	449	0	0	True	0.015731
	shgo-simplicial	2	34	2	2	True	0.001792
	shgo-sobol	2	31	1	1	True	0.001276

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	tgo	2	31	1	1	True	0.001143
GoldsteinPrice	bh	2	4587	0	0	True	0.050597
	de	2	781	0	0	True	0.021106
	shgo-simplicial	2	85	1	1	True	0.001664
	shgo-sobol	2	83	1	1	True	0.001773
	tgo	2	83	1	1	True	0.001681
Griewank	bh	2	1872	0	0	False	0.039220
	de	2	3283	0	0	True	0.124090
	shgo-simplicial	2	9	1	1	True	0.000633
	shgo-sobol	2	7	1	1	True	0.000710
	tgo	2	7	1	1	True	0.000604
Gulf	bh	3	404	0	0	False	0.025843
	de	3	15244	0	0	True	0.924735
	shgo-simplicial	3	650	3	3	True	0.050251
	shgo-sobol	3	234	1	1	True	0.010897
	tgo	3	234	1	1	True	0.010778
Hansen	bh	2	3432	0	0	True	0.104063
	de	2	1341	0	0	True	0.071333
	shgo-simplicial	2	130	3	3	True	0.004946
	shgo-sobol	2	114	1	1	True	0.004238
	tgo	2	379	7	7	True	0.014041
Hartmann3	bh	3	7464	0	0	True	0.132592
	de	3	720	0	0	True	0.026820
	shgo-simplicial	3	70	1	1	True	0.002384
	shgo-sobol	3	54	1	1	True	0.001875
	tgo	3	53	1	1	True	0.001753
Hartmann6	bh	6	16625	0	0	True	0.268922
	de	6	2230	0	0	True	0.091498
	shgo-simplicial	6	181	1	1	True	0.026658
	shgo-sobol	6	153	1	1	True	0.006292
	tgo	6	443	3	2	True	0.010686
HelicalValley	bh	3	7688	0	0	True	0.078989
	de	3	12124	0	0	True	0.361064
	shgo-simplicial	3	1456	4	4	True	0.129628

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	shgo-sobol	3	136	2	2	True	0.003249
	tgo	3	137	2	2	True	0.002614
HimmelBlau	bh	2	2529	0	0	True	0.023512
	de	2	4683	0	0	True	0.114841
	shgo-simplicial	2	66	1	1	True	0.001178
	shgo-sobol	2	45	1	1	True	0.001055
	tgo	2	45	1	1	True	0.000950
HolderTable	bh	2	1857	0	0	False	0.031476
	de	2	415	0	0	True	0.012619
	shgo-simplicial	2	179	2	2	True	0.010071
	shgo-sobol	2	97	2	2	True	0.002625
	tgo	2	117	3	3	True	0.002640
Hosaki	bh	2	2526	0	0	False	0.029150
	de	2	335	0	0	True	0.008496
	shgo-simplicial	2	47	1	1	True	0.001053
	shgo-sobol	2	29	1	1	True	0.000968
	tgo	2	29	1	1	True	0.000827
Infinity	bh	2	2583	0	0	True	0.040846
	de	2	3803	0	0	True	0.126713
	shgo-simplicial	2	13	1	1	True	0.001057
	shgo-sobol	2	150	1	1	True	0.004148
	tgo	2	121	1	1	True	0.002820
JennrichSampson	bh	2	10632	0	0	True	0.209818
	de	2	904	0	0	True	0.033887
	shgo-simplicial	2	52	1	1	True	0.001704
	shgo-sobol	2	50	1	1	True	0.001765
	tgo	2	50	1	1	True	0.001640
Judge	bh	2	3207	0	0	True	0.072541
	de	2	741	0	0	True	0.031098
	shgo-simplicial	2	53	1	1	True	0.001520
	shgo-sobol	2	51	1	1	True	0.001924
	tgo	2	51	1	1	True	0.001817
Katsuura	bh	2	474	0	0	False	0.025357
	de	2	2006	0	0	True	0.110219

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	shgo-simplicial	2	9	1	1	True	0.000852
	shgo-sobol	2	7	1	1	True	0.000788
	tgo	2	7	1	1	True	0.000699
Keane	bh	2	1566	0	0	True	0.023998
	de	2	7523	0	0	True	0.215796
	shgo-simplicial	2	1502	2	2	True	0.028949
	shgo-sobol	2	14	1	1	True	0.000787
	tgo	2	7	1	1	True	0.000507
Kowalik	bh	4	14660	0	0	True	0.248955
	de	4	6755	0	0	True	0.267475
	shgo-simplicial	4	240	1	1	True	0.006430
	shgo-sobol	4	229	1	1	True	0.005653
	tgo	4	228	1	1	True	0.005533
Langermann	bh	2	2877	0	0	True	0.098686
	de	2	692	0	0	True	0.035666
	shgo-simplicial	2	397	5	5	True	0.023237
	shgo-sobol	2	49	1	1	True	0.002482
	tgo	2	49	1	1	True	0.002274
LennardJones	bh	6	10374	0	0	True	0.058650
	de	6	15748	0	0	True	0.451860
	shgo-simplicial	6	124	1	1	True	0.001939
	shgo-sobol	6	81	1	1	True	0.002031
	tgo	6	173	1	1	True	0.002409
Leon	bh	2	6207	0	0	True	0.042320
	de	2	5363	0	0	True	0.129094
	shgo-simplicial	2	99	1	1	True	0.001523
	shgo-sobol	2	97	1	1	True	0.001644
	tgo	2	97	1	1	True	0.001533
Levy03	bh	2	2670	0	0	False	0.067339
	de	2	3803	0	0	True	0.163955
	shgo-simplicial	2	45	1	1	True	0.001729
	shgo-sobol	2	43	1	1	True	0.001749
	tgo	2	43	1	1	True	0.001647
Levy13	bh	2	4491	0	0	True	0.053878

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	de	2	3803	0	0	True	0.114689
	shgo-simplicial	2	20	1	1	True	0.000724
	shgo-sobol	2	18	1	1	True	0.000825
	tgo	2	18	1	1	True	0.000704
Matyas	bh	2	1803	0	0	True	0.018082
	de	2	4323	0	0	True	0.103091
	shgo-simplicial	2	9	1	1	True	0.000461
	shgo-sobol	2	7	1	1	True	0.000587
	tgo	2	7	1	1	True	0.000470
McCormick	bh	2	2073	0	0	False	0.023725
	de	2	495	0	0	True	0.012853
	shgo-simplicial	2	42	1	1	True	0.000948
	shgo-sobol	2	40	1	1	True	0.001059
	tgo	2	40	1	1	True	0.000956
Michalewicz	bh	2	4320	0	0	True	0.070726
	de	2	498	0	0	True	0.017048
	shgo-simplicial	2	50	1	1	True	0.001517
	shgo-sobol	2	48	1	1	True	0.001593
	tgo	2	48	1	1	True	0.001480
MieleCantrell	bh	4	9270	0	0	True	0.084157
	de	4	42965	0	0	True	1.297974
	shgo-simplicial	4	455	1	1	True	0.007732
	shgo-sobol	4	444	1	1	True	0.006351
	tgo	4	443	1	1	True	0.006228
Mishra01	bh	2	1830	0	0	False	0.025162
	de	2	406	0	0	True	0.011320
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	11	1	1	True	0.000743
	tgo	2	11	1	1	True	0.000581
Mishra02	bh	2	1752	0	0	False	0.028230
	de	2	566	0	0	True	0.017230
	shgo-simplicial	2	9	1	1	True	0.000513
	shgo-sobol	2	0	0	0	False	0.000000
	tgo	2	0	0	0	False	0.000000

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
Mishra03	bh	2	21105	0	0	False	0.191135
	de	2	2028	0	0	False	0.057982
	shgo-simplicial	2	70	1	1	True	0.001465
	shgo-sobol	2	68	1	1	True	0.001578
	tgo	2	68	1	1	True	0.001521
Mishra04	bh	2	23679	0	0	False	0.212897
	de	2	1663	0	0	False	0.048371
	shgo-simplicial	2	599	3	3	True	0.016789
	shgo-sobol	2	4357	8	8	True	0.072626
	tgo	2	9363	18	18	True	0.149810
Mishra05	bh	2	7512	0	0	False	0.100622
	de	2	852	0	0	False	0.026962
	shgo-simplicial	2	50	1	1	True	0.001769
	shgo-sobol	2	142	2	2	True	0.003643
	tgo	2	263	3	3	True	0.005386
Mishra06	bh	2	2346	0	0	False	0.042102
	de	2	695	0	0	True	0.022795
	shgo-simplicial	2	62	1	1	True	0.002086
	shgo-sobol	2	121	2	2	True	0.003196
	tgo	2	170	2	2	True	0.003860
Mishra07	bh	2	1230	0	0	True	0.028053
	de	2	7043	0	0	True	0.250454
	shgo-simplicial	2	170	1	1	True	0.024871
	shgo-sobol	2	47	2	2	True	0.001797
	tgo	2	84	3	3	True	0.002268
Mishra08	bh	2	9831	0	0	True	0.122770
	de	2	2903	0	0	True	0.092307
	shgo-simplicial	2	243	1	1	True	0.004905
	shgo-sobol	2	235	1	1	True	0.005008
	tgo	2	235	1	1	True	0.004814
Mishra10	bh	2	303	0	0	False	0.010973
	de	2	923	0	0	True	0.021494
	shgo-simplicial	2	9	1	1	True	0.000476
	shgo-sobol	2	7	1	1	True	0.000583

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	tgo	2	7	1	1	True	0.000472
Mishra11	bh	2	1140	0	0	True	0.022872
	de	2	6443	0	0	True	0.207037
	shgo-simplicial	2	9	1	1	True	0.000561
	shgo-sobol	2	7	1	1	True	0.000644
	tgo	2	7	1	1	True	0.000520
MultiModal	bh	2	21084	0	0	True	0.192647
	de	2	3643	0	0	True	0.111857
	shgo-simplicial	2	9	1	1	True	0.000544
	shgo-sobol	2	7	1	1	True	0.000635
	tgo	2	7	1	1	True	0.000513
NeedleEye	bh	2	10005	0	0	True	0.088015
	de	2	247	0	0	True	0.004802
	shgo-simplicial	2	9	1	1	True	0.000515
	shgo-sobol	2	7	1	1	True	0.000609
	tgo	2	7	1	1	True	0.000502
NewFunction01	bh	2	20874	0	0	False	0.170837
	de	2	1683	0	0	False	0.045641
	shgo-simplicial	2	3813	6	6	True	0.064411
	shgo-sobol	2	1569	6	6	True	0.026975
	tgo	2	10079	23	23	True	0.155735
NewFunction02	bh	2	22662	0	0	False	0.186096
	de	2	1721	0	0	False	0.045242
	shgo-simplicial	2	159	1	1	True	0.004481
	shgo-sobol	2	341	2	2	True	0.005782
	tgo	2	361	2	2	True	0.005829
OddSquare	bh	2	303	0	0	False	0.015612
	de	2	1238	0	0	True	0.042061
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	204	1	1	True	0.006156
	tgo	2	487	8	8	True	0.012439
Parsopoulos	bh	2	1608	0	0	True	0.020883
	de	2	4163	0	0	True	0.110717
	shgo-simplicial	2	30	1	1	True	0.001201

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	shgo-sobol	2	39	1	1	True	0.001080
	tgo	2	39	1	1	True	0.000929
Pathological	bh	2	8106	0	0	True	0.192009
	de	2	2498	0	0	True	0.109880
	shgo-simplicial	2	20	1	1	True	0.000988
	shgo-sobol	2	18	1	1	True	0.001041
	tgo	2	18	1	1	True	0.001183
Paviani	bh	10	13970	0	0	False	0.231402
	de	10	6088	0	0	True	0.261439
	shgo-simplicial	10	1257	1	1	True	180.467203
	shgo-sobol	10	364	1	1	True	0.013044
	tgo	10	358	1	1	True	0.007403
PenHolder	bh	2	1512	0	0	False	0.029009
	de	2	532	0	0	True	0.017184
	shgo-simplicial	2	117	2	2	True	0.004500
	shgo-sobol	2	86	2	2	True	0.002352
	tgo	2	74	2	2	True	0.001852
Penalty01	bh	2	3300	0	0	True	0.103837
	de	2	3803	0	0	True	0.192800
	shgo-simplicial	2	45	1	1	True	0.002049
	shgo-sobol	2	43	1	1	True	0.002108
	tgo	2	43	1	1	True	0.001994
PermFunction01	bh	2	4599	0	0	True	0.132834
	de	2	4963	0	0	True	0.250583
	shgo-simplicial	2	83	1	1	True	0.003318
	shgo-sobol	2	70	1	1	True	0.002988
	tgo	2	70	1	1	True	0.002867
PermFunction02	bh	2	4665	0	0	True	0.130644
	de	2	4563	0	0	True	0.228565
	shgo-simplicial	2	73	1	1	True	0.002946
	shgo-sobol	2	71	1	1	True	0.002968
	tgo	2	71	1	1	True	0.002847
Pinter	bh	2	3075	0	0	False	0.121511
	de	2	4043	0	0	True	0.230878

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	shgo-simplicial	2	9	1	1	True	0.000793
	shgo-sobol	2	7	1	1	True	0.000814
	tgo	2	7	1	1	True	0.000695
Plateau	bh	2	303	0	0	False	0.012533
	de	2	283	0	0	True	0.007515
	shgo-simplicial	2	9	1	1	True	0.000512
	shgo-sobol	2	7	1	1	True	0.000612
	tgo	2	7	1	1	True	0.000497
Powell	bh	4	14285	0	0	True	0.090999
	de	4	35285	0	0	True	0.939842
	shgo-simplicial	4	220	1	1	True	0.003550
	shgo-sobol	4	209	1	1	True	0.002965
	tgo	4	208	1	1	True	0.002803
PowerSum	bh	4	53785	0	0	True	1.136949
	de	4	80125	0	0	True	3.736967
	shgo-simplicial	4	386	1	1	True	0.011866
	shgo-sobol	4	641	1	1	True	0.018445
	tgo	4	640	1	1	True	0.018284
Price01	bh	2	921	0	0	True	0.016215
	de	2	4003	0	0	True	0.106303
	shgo-simplicial	2	14	1	1	True	0.000571
	shgo-sobol	2	12	1	1	True	0.000679
	tgo	2	12	1	1	True	0.000559
Price02	bh	2	1614	0	0	False	0.028523
	de	2	732	0	0	False	0.023422
	shgo-simplicial	2	9	1	1	True	0.000519
	shgo-sobol	2	7	1	1	True	0.000657
	tgo	2	7	1	1	True	0.000557
Price03	bh	2	5136	0	0	True	0.040620
	de	2	4283	0	0	True	0.107037
	shgo-simplicial	2	58	1	1	True	0.001118
	shgo-sobol	2	56	1	1	True	0.001240
	tgo	2	56	1	1	True	0.001102
Price04	bh	2	6984	0	0	True	0.050386

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	de	2	40043	0	0	True	1.001060
	shgo-simplicial	2	9	1	1	True	0.000480
	shgo-sobol	2	7	1	1	True	0.000593
	tgo	2	7	1	1	True	0.000484
Quadratic	bh	2	1917	0	0	True	0.019071
	de	2	378	0	0	True	0.008401
	shgo-simplicial	2	35	1	1	True	0.000803
	shgo-sobol	2	33	1	1	True	0.000902
	tgo	2	33	1	1	True	0.000795
Quintic	bh	2	35934	0	0	True	0.653358
	de	2	3823	0	0	True	0.155548
	shgo-simplicial	2	114	1	1	True	0.003436
	shgo-sobol	2	112	1	1	True	0.003479
	tgo	2	112	1	1	True	0.003377
Rana	bh	2	1851	0	0	False	0.044188
	de	2	1055	0	0	False	0.041408
	shgo-simplicial	2	292	5	5	True	0.010485
	shgo-sobol	2	651	10	10	True	0.019660
	tgo	2	1157	20	20	True	0.032245
Rastrigin	bh	2	3198	0	0	True	0.045106
	de	2	2323	0	0	True	0.075224
	shgo-simplicial	2	20	1	1	True	0.000758
	shgo-sobol	2	18	1	1	True	0.000832
	tgo	2	18	1	1	True	0.000722
Ratkowsky01	bh	4	5355	0	0	False	0.108619
	de	4	3595	0	0	False	0.147932
	shgo-simplicial	4	286	1	1	True	0.008561
	shgo-sobol	4	187	1	1	True	0.005141
	tgo	4	186	1	1	True	0.005016
Ratkowsky02	bh	3	18628	0	0	True	0.325850
	de	3	2308	0	0	True	0.089465
	shgo-simplicial	3	124	1	1	True	0.005340
	shgo-sobol	3	111	1	1	True	0.002979
	tgo	3	110	1	1	True	0.002856

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
Ripple01	bh	2	5961	0	0	False	0.124923
	de	2	824	0	0	False	0.033875
	shgo-simplicial	2	153	3	3	True	0.006820
	shgo-sobol	2	476	1	1	True	0.016043
	tgo	2	1879	29	29	True	0.061017
Ripple25	bh	2	3426	0	0	False	0.068602
	de	2	815	0	0	True	0.030853
	shgo-simplicial	2	106	3	3	True	0.005244
	shgo-sobol	2	99	1	1	True	0.003343
	tgo	2	372	9	9	True	0.009879
Rosenbrock	bh	2	6324	0	0	True	0.100650
	de	2	4923	0	0	True	0.173339
	shgo-simplicial	2	118	1	1	True	0.002990
	shgo-sobol	2	116	1	1	True	0.003024
	tgo	2	116	1	1	True	0.002962
RosenbrockModified	bh	2	5790	0	0	False	0.058106
	de	2	898	0	0	True	0.024119
	shgo-simplicial	2	128	2	2	True	0.004098
	shgo-sobol	2	66	1	1	True	0.001674
	tgo	2	66	1	1	True	0.001431
RotatedEllipse01	bh	2	1566	0	0	True	0.019852
	de	2	3923	0	0	True	0.101885
	shgo-simplicial	2	9	1	1	True	0.000527
	shgo-sobol	2	7	1	1	True	0.000608
	tgo	2	7	1	1	True	0.000489
RotatedEllipse02	bh	2	1542	0	0	True	0.016783
	de	2	3763	0	0	True	0.091090
	shgo-simplicial	2	9	1	1	True	0.000463
	shgo-sobol	2	7	1	1	True	0.000584
	tgo	2	7	1	1	True	0.000468
Salomon	bh	2	7743	0	0	True	0.090562
	de	2	1489	0	0	False	0.048057
	shgo-simplicial	2	40	1	1	True	0.001145
	shgo-sobol	2	38	1	1	True	0.001219

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	tgo	2	38	1	1	True	0.001105
Sargan	bh	2	1536	0	0	True	0.062461
	de	2	4003	0	0	True	0.234294
	shgo-simplicial	2	9	1	1	True	0.000796
	shgo-sobol	2	7	1	1	True	0.000806
	tgo	2	7	1	1	True	0.000697
Schaffer01	bh	2	3273	0	0	False	0.030547
	de	2	2883	0	0	True	0.076986
	shgo-simplicial	2	9	1	1	True	0.000513
	shgo-sobol	2	7	1	1	True	0.000602
	tgo	2	7	1	1	True	0.000492
Schaffer02	bh	2	4806	0	0	False	0.043977
	de	2	2803	0	0	True	0.077345
	shgo-simplicial	2	9	1	1	True	0.000485
	shgo-sobol	2	7	1	1	True	0.000590
	tgo	2	7	1	1	True	0.000489
Schaffer03	bh	2	6183	0	0	False	0.065953
	de	2	3289	0	0	True	0.094391
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	870	2	2	True	0.014601
	tgo	2	6986	15	15	True	0.112689
Schaffer04	bh	2	4956	0	0	False	0.054692
	de	2	1769	0	0	True	0.051488
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	956	2	2	True	0.016079
	tgo	2	7424	16	16	True	0.120559
Schwefel01	bh	2	2592	0	0	True	0.034659
	de	2	4003	0	0	True	0.119275
	shgo-simplicial	2	9	1	1	True	0.002122
	shgo-sobol	2	7	1	1	True	0.000628
	tgo	2	7	1	1	True	0.000508
Schwefel02	bh	2	1530	0	0	True	0.051924
	de	2	4563	0	0	True	0.231409
	shgo-simplicial	2	9	1	1	True	0.000774

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	shgo-sobol	2	7	1	1	True	0.000781
	tgo	2	7	1	1	True	0.000653
Schwefel04	bh	2	2856	0	0	True	0.047694
	de	2	4403	0	0	True	0.148907
	shgo-simplicial	2	47	1	1	True	0.001346
	shgo-sobol	2	45	1	1	True	0.001410
	tgo	2	45	1	1	True	0.001343
Schwefel06	bh	2	32622	0	0	True	0.224756
	de	2	4263	0	0	True	0.114865
	shgo-simplicial	2	150	1	1	True	0.002512
	shgo-sobol	2	148	1	1	True	0.002586
	tgo	2	148	1	1	True	0.002469
Schwefel20	bh	2	37062	0	0	True	0.251657
	de	2	3663	0	0	True	0.102275
	shgo-simplicial	2	72	1	1	True	0.001425
	shgo-sobol	2	70	1	1	True	0.001518
	tgo	2	70	1	1	True	0.001410
Schwefel21	bh	2	30786	0	0	True	0.151423
	de	2	4263	0	0	True	0.103540
	shgo-simplicial	2	23	1	1	True	0.000652
	shgo-sobol	2	21	1	1	True	0.000780
	tgo	2	21	1	1	True	0.000653
Schwefel22	bh	2	34536	0	0	True	0.315124
	de	2	3903	0	0	True	0.119683
	shgo-simplicial	2	75	1	1	True	0.001700
	shgo-sobol	2	73	1	1	True	0.001798
	tgo	2	73	1	1	True	0.001667
Schwefel26	bh	2	1377	0	0	False	0.023562
	de	2	1763	0	0	True	0.059069
	shgo-simplicial	2	85	2	2	True	0.060432
	shgo-sobol	2	46	1	1	True	0.001913
	tgo	2	46	1	1	True	0.001716
Schwefel36	bh	2	531	0	0	False	0.012828
	de	2	741	0	0	True	0.017205

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	shgo-simplicial	2	593	1	1	True	0.029588
	shgo-sobol	2	514	1	1	True	0.010140
	tgo	2	80	2	2	True	0.001428
Shekel05	bh	4	4765	0	0	False	0.077725
	de	4	2505	0	0	True	0.095390
	shgo-simplicial	4	118	1	1	True	0.003752
	shgo-sobol	4	107	1	1	True	0.002920
	tgo	4	106	1	1	True	0.002766
Shekel07	bh	4	5720	0	0	True	0.091560
	de	4	2590	0	0	True	0.099127
	shgo-simplicial	4	134	1	1	True	0.004098
	shgo-sobol	4	123	1	1	True	0.003263
	tgo	4	122	1	1	True	0.003110
Shekel10	bh	4	4365	0	0	False	0.073112
	de	4	2500	0	0	False	0.097044
	shgo-simplicial	4	142	1	1	True	0.004259
	shgo-sobol	4	131	1	1	True	0.003466
	tgo	4	130	1	1	True	0.003383
Shubert01	bh	2	3111	0	0	True	0.067104
	de	2	1467	0	0	True	0.061174
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	76	1	1	True	0.003538
	tgo	2	157	3	3	True	0.005496
Shubert03	bh	2	3000	0	0	True	0.069237
	de	2	1055	0	0	True	0.045163
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	51	1	1	True	0.002095
	tgo	2	51	1	1	True	0.001756
Shubert04	bh	2	3024	0	0	True	0.069495
	de	2	1175	0	0	True	0.049131
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	142	3	3	True	0.004916
	tgo	2	178	4	4	True	0.005449
SineEnvelope	bh	2	1416	0	0	False	0.034150

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	de	2	1449	0	0	False	0.053516
	shgo-simplicial	2	9	1	1	True	0.000597
	shgo-sobol	2	7	1	1	True	0.000680
	tgo	2	7	1	1	True	0.000574
SixHumpCamel	bh	2	3030	0	0	True	0.028893
	de	2	615	0	0	True	0.014982
	shgo-simplicial	2	175	1	1	True	0.009341
	shgo-sobol	2	42	1	1	True	0.001182
	tgo	2	42	1	1	True	0.000968
Sodp	bh	2	3648	0	0	True	0.051512
	de	2	4043	0	0	True	0.130966
	shgo-simplicial	2	9	1	1	True	0.000509
	shgo-sobol	2	7	1	1	True	0.000609
	tgo	2	7	1	1	True	0.000515
Sphere	bh	2	909	0	0	True	0.017210
	de	2	3603	0	0	True	0.101877
	shgo-simplicial	2	9	1	1	True	0.000493
	shgo-sobol	2	7	1	1	True	0.000607
	tgo	2	7	1	1	True	0.000493
Step	bh	2	303	0	0	False	0.012524
	de	2	1083	0	0	True	0.030122
	shgo-simplicial	2	9	1	1	True	0.000481
	shgo-sobol	2	7	1	1	True	0.000605
	tgo	2	7	1	1	True	0.000497
Step2	bh	2	303	0	0	False	0.013313
	de	2	843	0	0	True	0.025589
	shgo-simplicial	2	9	1	1	True	0.000516
	shgo-sobol	2	7	1	1	True	0.000630
	tgo	2	7	1	1	True	0.000509
StretchedV	bh	2	1866	0	0	True	0.039634
	de	2	1529	0	0	True	0.055269
	shgo-simplicial	2	43	1	1	True	0.001558
	shgo-sobol	2	41	1	1	True	0.001483
	tgo	2	41	1	1	True	0.001373

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
StyblinskiTang	bh	2	2031	0	0	False	0.036926
	de	2	492	0	0	True	0.015949
	shgo-simplicial	2	41	1	1	True	0.001222
	shgo-sobol	2	48	1	1	True	0.001570
	tgo	2	48	1	1	True	0.001405
TestTubeHolder	bh	2	1563	0	0	False	0.027345
	de	2	852	0	0	True	0.025876
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	895	1	1	True	0.024863
	tgo	2	1101	31	30	True	0.023859
ThreeHumpCamel	bh	2	2247	0	0	True	0.022462
	de	2	4163	0	0	True	0.103089
	shgo-simplicial	2	9	1	1	True	0.000438
	shgo-sobol	2	7	1	1	True	0.000586
	tgo	2	7	1	1	True	0.000479
Treccani	bh	2	2658	0	0	True	0.023854
	de	2	2403	0	0	True	0.062638
	shgo-simplicial	2	9	1	1	True	0.000466
	shgo-sobol	2	7	1	1	True	0.000592
	tgo	2	7	1	1	True	0.000476
Trid	bh	6	9387	0	0	True	0.115005
	de	6	4178	0	0	True	0.146371
	shgo-simplicial	6	152	1	1	True	0.025636
	shgo-sobol	6	98	1	1	True	0.002638
	tgo	6	94	1	1	True	0.002139
Trigonometric01	bh	2	6288	0	0	True	0.149112
	de	2	6843	0	0	True	0.316741
	shgo-simplicial	2	9	1	1	True	0.000654
	shgo-sobol	2	7	1	1	True	0.000729
	tgo	2	7	1	1	True	0.000627
Tripod	bh	2	27252	0	0	False	0.200039
	de	2	3863	0	0	True	0.111605
	shgo-simplicial	2	163	2	2	True	0.004474
	shgo-sobol	2	254	2	2	True	0.004403

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	tgo	2	254	2	2	True	0.004049
Ursem01	bh	2	1911	0	0	False	0.023793
	de	2	332	0	0	True	0.008287
	shgo-simplicial	2	22	1	1	True	0.000700
	shgo-sobol	2	29	1	1	True	0.000926
	tgo	2	29	1	1	True	0.000811
Ursem03	bh	2	5286	0	0	False	0.072546
	de	2	782	0	0	True	0.019909
	shgo-simplicial	2	55	1	1	True	0.001412
	shgo-sobol	2	53	1	1	True	0.001487
	tgo	2	53	1	1	True	0.001377
Ursem04	bh	2	16347	0	0	True	0.138740
	de	2	591	0	0	True	0.013742
	shgo-simplicial	2	97	1	1	True	0.001742
	shgo-sobol	2	95	1	1	True	0.001855
	tgo	2	95	1	1	True	0.001740
UrsemWaves	bh	2	420	0	0	False	0.014496
	de	2	498	0	0	False	0.013866
	shgo-simplicial	2	13	2	2	True	0.000739
	shgo-sobol	2	19	1	1	True	0.000854
	tgo	2	19	1	1	True	0.000707
VSS	bh	2	2448	0	0	False	0.034364
	de	2	655	0	0	True	0.019093
	shgo-simplicial	2	9	1	1	True	0.000497
	shgo-sobol	2	7	1	1	True	0.000617
	tgo	2	7	1	1	True	0.000507
Vincent	bh	2	2805	0	0	True	0.056584
	de	2	753	0	0	True	0.021509
	shgo-simplicial	2	42	1	1	True	0.001052
	shgo-sobol	2	31	1	1	True	0.001015
	tgo	2	31	1	1	True	0.000920
Watson	bh	6	33320	0	0	True	1.415519
	de	6	23095	0	0	True	1.642898
	shgo-simplicial	6	337	1	1	True	0.042924

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	shgo-sobol	6	283	1	1	True	0.016899
	tgo	6	279	1	1	True	0.015085
Wavy	bh	2	3465	0	0	False	0.054129
	de	2	2603	0	0	True	0.089450
	shgo-simplicial	2	9	1	1	True	0.000559
	shgo-sobol	2	7	1	1	True	0.000659
	tgo	2	7	1	1	True	0.000548
WayburnSeader01	bh	2	6933	0	0	True	0.052945
	de	2	4823	0	0	True	0.127303
	shgo-simplicial	2	111	1	1	True	0.001713
	shgo-sobol	2	109	1	1	True	0.001851
	tgo	2	109	1	1	True	0.001741
WayburnSeader02	bh	2	6732	0	0	True	0.055006
	de	2	5043	0	0	True	0.126818
	shgo-simplicial	2	150	1	1	True	0.002188
	shgo-sobol	2	148	1	1	True	0.002326
	tgo	2	148	1	1	True	0.002210
Weierstrass	bh	2	30213	0	0	False	1.013885
	de	2	3623	0	0	True	0.210147
	shgo-simplicial	2	2225	1	1	True	0.218606
	shgo-sobol	2	0	0	0	False	0.000000
	tgo	2	0	0	0	False	0.000000
Whitley	bh	2	5244	0	0	False	0.123444
	de	2	1618	0	0	False	0.071220
	shgo-simplicial	2	34	1	1	True	0.001424
	shgo-sobol	2	32	1	1	True	0.001498
	tgo	2	32	1	1	True	0.001402
Wolfe	bh	3	1156	0	0	False	0.015604
	de	3	16444	0	0	True	0.422252
	shgo-simplicial	3	14	1	1	True	0.001167
	shgo-sobol	3	10	1	1	True	0.000682
	tgo	3	9	1	1	True	0.000555
XinSheYang01	bh	2	7632	0	0	False	0.097735
	de	2	6663	0	0	True	0.220974

Continued on next page

Problem	Alg	ndim	nfev	nlmin	nulmin	success	runtime
	shgo-simplicial	2	153	1	1	True	0.003723
	shgo-sobol	2	77	1	1	True	0.002016
	tgo	2	149	1	1	True	0.003290
XinSheYang02	bh	2	2691	0	0	False	0.046717
	de	2	4223	0	0	True	0.141718
	shgo-simplicial	2	65	1	1	True	0.001683
	shgo-sobol	2	63	1	1	True	0.001770
	tgo	2	63	1	1	True	0.001675
XinSheYang03	bh	2	312	0	0	False	0.017095
	de	2	1409	0	0	True	0.058742
	shgo-simplicial	2	9	1	1	True	0.000640
	shgo-sobol	2	7	1	1	True	0.000714
	tgo	2	7	1	1	True	0.000594
XinSheYang04	bh	2	1791	0	0	False	0.044684
	de	2	1795	0	0	True	0.065124
	shgo-simplicial	2	77	1	1	True	0.002366
	shgo-sobol	2	75	1	1	True	0.002478
	tgo	2	75	1	1	True	0.002320
Xor	bh	9	7940	0	0	False	0.207127
	de	9	1520	0	0	False	0.061928
	shgo-simplicial	9	645	1	1	True	15.690898
	shgo-sobol	9	197	1	1	True	0.017347
	tgo	9	208	2	2	True	0.006470
YaoLiu04	bh	2	29316	0	0	True	0.173835
	de	2	3903	0	0	True	0.100743
	shgo-simplicial	2	23	1	1	True	0.000677
	shgo-sobol	2	21	1	1	True	0.000803
	tgo	2	21	1	1	True	0.000677
YaoLiu09	bh	2	3300	0	0	True	0.049375
	de	2	2843	0	0	True	0.093783
	shgo-simplicial	2	20	1	1	True	0.000767
	shgo-sobol	2	18	1	1	True	0.000867
	tgo	2	18	1	1	True	0.000758
Zacharov	bh	2	2046	0	0	True	0.039036

Continued on next page

		ndim	nfev	nlmin	nulmin	success	runtime
Problem	Alg						
	de	2	4043	0	0	True	0.143251
	shgo-simplicial	2	46	1	1	True	0.001841
	shgo-sobol	2	45	1	1	True	0.001503
	tgo	2	45	1	1	True	0.001399
ZeroSum	bh	2	20538	0	0	False	0.245236
	de	2	1743	0	0	False	0.056568
	shgo-simplicial	2	0	0	0	False	0.000000
	shgo-sobol	2	23	1	1	True	0.001420
	tgo	2	0	0	0	False	0.000000
Zettl	bh	2	4167	0	0	True	0.033280
	de	2	861	0	0	True	0.020092
	shgo-simplicial	2	116	1	1	True	0.001764
	shgo-sobol	2	114	1	1	True	0.001909
	tgo	2	114	1	1	True	0.001793
Zimmerman	bh	2	24543	0	0	False	0.277145
	de	2	6503	0	0	True	0.207111
	shgo-simplicial	2	3032	1	1	True	0.173071
	shgo-sobol	2	1585	1	1	True	0.033359
	tgo	2	1585	1	1	True	0.041850
Zirilli	bh	2	2562	0	0	False	0.023889
	de	2	575	0	0	True	0.013579
	shgo-simplicial	2	34	1	1	True	0.000779
	shgo-sobol	2	32	1	1	True	0.000948
	tgo	2	32	1	1	True	0.000811

Bibliography

Adorio, E. P. and Dilman, U. P. "MVF - Multivariate Test Functions Library in C for Unconstrained Global Optimization", (2005).

Antonov, I. A. and Saleev, V. M. (1979) "An economic method of computing LP-sequences", *USSR Comput. Math. Math. Phys., 19*, 252–256.

Atanassov, K. (1996) "On sperner's lemma", *Studia Scientiarum Mathematicarum Hungarica, 32* (1), 71–74.

Barber, C. B. and Dobkin, D. P. (1996) "The Quickhull Algorithm for Convex Hulls", *22* (4), 469–483.

Bigoni, D. "UQToolbox 1.0.3 Tools for Uncertainty Quantification", (2016).

Brouwer, L. E. J. (1911) "Über Abbildung von Mannigfaltigkeiten", *Mathematische Annalen, 71* (1), 97–115 URL `http://dx.doi.org/10.1007/BF01456931`.

De Loera, J. A.; Peterson, E. and Edward Su, F. (2002) "A Polytopal Generalization of Sperner's Lemma", *Journal of Combinatorial Theory, Series A, 100* (1), 1–26 URL `http://linkinghub.elsevier.com/retrieve/pii/S0097316502932747`.

Dolan, E. D. and Moré, J. J. Jan (2002) "Benchmarking optimization software with performance profiles", *Mathematical Programming, 91* (2), 201–213 ISSN 1436-4646 URL `https://doi.org/10.1007/s101070100263`.

Eilenberg, S. and Steenrod, N. (1952) "Foundations of algebraic topology", *Mathematical Reviews (MathSciNet): MR14: 398b Zentralblatt MATH, Princeton, 47*.

Endres, S. "SHGO: Python implementation of the simplicial homology global optimisation algorithm", (2016–)a URL `https://bitbucket.org/upiamcompthermo/shgo`.

Endres, S. "TGO: Python implementation of the topograhphical global optimisation algorithm", (2016–)b URL `https://bitbucket.org/account/user/upiamcompthermo/projects/TGO`.

Finkel, D. E. (2003) “Direct optimization algorithm user guide”, *Center for Research in Scientific Computation, North Carolina State University, 2*.

Gavana, A. “ Global Optimization Benchmarks and AMPGO”, (2016).

Hatcher, A. (2002) *Algebraic topology*, Cambridge University Press, Cambridge ISBN 0-521-79160-X; 0-521-79540-0.

Henderson, N.; de Sá Rêgo, M. and Imbiriba, J. (2017) “Topographical global initialization for finding all solutions of nonlinear systems with constraints”, *Applied Numerical Mathematics, 112*, 155 – 166 URL `http://www.sciencedirect.com/science/article/pii/S016892741630201X`.

Henderson, N.; de Sá Rêgo, M.; Sacco, W. F. and Rodrigues, R. A. (2015) “A new look at the topographical global optimization method and its application to the phase stability analysis of mixtures”, *Chemical Engineering Science, 127*, 151–174 URL `http://linkinghub.elsevier.com/retrieve/pii/S0009250915000494`.

Henle, M. (1979) *A Combinatorial Introduction to Topology*, Unabriged Dover (1994) republication of the edition published by WH Greeman & Company, San Francisco, 1979, .

Herskovits, J. (1998) “Feasible Direction Interior-Point Technique”, *99* (1), 121–146.

Huyer, W. and Neumaier, A. Jun (1999) “Global optimization by multilevel coordinate search”, *Journal of Global Optimization, 14* (4), 331–355 ISSN 1573-2916 URL `https://doi.org/10.1023/A:1008382309369`.

Jamil, M. and Yang, X.-S. (2013) “A Literature Survey of Benchmark Functions For Global Optimization Problems Citation details: Momin Jamil and Xin-She Yang, A literature survey of benchmark functions for global optimization problems”, *Int. Journal of Mathematical Modelling and Numerical Optimisation, 4* (2), 150–194.

Jones, D. R.; Perttunen, C. D. and Stuckman, B. E. Oct (1993) “Lipschitzian optimization without the lipschitz constant”, *Journal of Optimization Theory and Applications, 79* (1), 157–181.

Jones, E.; Oliphant, T.; Peterson, P. et al. “SciPy: Open source scientific tools for Python”, (2001–) URL `http://www.scipy.org/`.

Keenan Crane, Fernando de Goes, M. D. P. S. “Digital geometry processing with discrete exterior calculus”, in *ACM SIGGRAPH 2013 courses*, SIGGRAPH ’13 New York, NY, USA (2013) ACM.

Kraft, D. (1988) “A software package for sequential quadratic programming”, *Technical Report DFVLR-FB 88-28, Institut fuer Dynamik der Flugsysteme, Oberpfaffenhofen,.*

Kraft, D. (1994) “Algorithm 733: TOMP-Fortran modules for optimal control calculations”, *ACM Transactions on Mathematical Software, 20* (3), 262–281.

Kuipers, L. and Niederreiter, H. (1974) *Uniform distribution of sequences.*, page 192.

Li, Z. and Scheraga, H. A. (1987) “Monte carlo-minimization approach to the multiple-minima problem in protein folding”, *Proceedings of the National Academy of Sciences, 84* (19), 6611–6615.

Meunier, F. (2006) “Sperner labellings: A combinatorial approach”, *Journal of Combinatorial Theory, Series A, 113* (7), 1462 – 1475 URL `http://www.sciencedirect.com/science/article/pii/S0097316506000094`.

Mishra, S. “Some new test functions for global optimization and performance of repulsive particle swarm method”, (2007) URL `http://mpra.ub.uni-muenchen.de/2718/`.

Mishra, S. K. “Global Optimization by Differential Evolution and Particle Swarm Methods Evaluation on Some Benchmark Functions”, (2006).

Musin, O. R. (2015) “Extensions of Sperner and Tucker’s lemma”, *Journal of Combinatorial Theory, Series A, 132*, 172–187 URL `http://dx.doi.org/10.1016/j.jcta.2014.12.001`.

NIST “NIST StRD Nonlinear Regression Problems”, (2016).

Paulavičius, R.; Sergeyev, Y. D.; Kvasov, D. E. and Žilinskas, J. Jul (2014) “Globally-biased disimpl algorithm for expensive global optimization”, *Journal of Global Optimization, 59* (2), 545–567.

Paulavičius, R. and Žilinskas, J. (2014)a *Simplicial global optimization*, Springer, .

Paulavičius, R. and Žilinskas, J. May (2014)b “Simplicial lipschitz optimization without the lipschitz constant”, *Journal of Global Optimization, 59* (1), 23–40.

Paulavičius, R. and Žilinskas, J. Feb (2016) “Advantages of simplicial partitioning for lipschitz optimization problems with linear constraints”, *Optimization Letters, 10* (2), 237–246.

Rios, L. M. and Sahinidis, N. V. Jul (2013) “Derivative-free optimization: a review of algorithms and comparison of software implementations”, *Journal of Global Optimization, 56* (3), 1247–1293.

Shan, S. and Wang, G. G. (2010) "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions", *Structural and Multidisciplinary Optimization, 41* (2), 219–241 URL `http://dx.doi.org/10.1007/s00158-009-0420-2`.

Sobol, I. M. (1967) "The distribution of points in a cube and the approximate evaluation of integrals", *USSR Comput. Math. Math. Phys., 7*, 86–112.

Sperner, E. (1928) "Neuer beweis für die invarianz der dimensionszahl und des gebietes", *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, 6* (1), 265.

Storn, R. and Price, K. (1997) "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization, 11* (4), 341–359 URL `http://dx.doi.org/10.1023/A:1008202821328`.

Törn, A. (1990) "Topographical global optimization", *Reports on Computer Science and Mathematics, No 199*.

Törn, A. "Clustering methods in global optimization, (in: Preprints of the second ifac symposium on stochastic control, sopron, hungary, part 2)", pages 138–143 (1986).

Törn, A. and Viitanen, S. (1992) *Topographical Global Optimization, (in Recent Advances in Global Optimization)*, pages 384–398 Princeton University Press Princeton, NJ.

Törn, A. and Viitanen, S. (1996) *Iterative Topographical Global Optimization*, pages 353–363 Springer US Boston, MA ISBN 978-1-4613-3437-8 URL `http://dx.doi.org/10.1007/978-1-4613-3437-8_22`.

Vaz, A. I. and Vicente, L. N. (2009) "Pswarm: a hybrid solver for linearly constrained global derivative-free optimization", *Optimization Methods and Software, 24* (4-5), 669–685 URL `http://dx.doi.org/10.1080/10556780902909948`.

Wales, D. J. (2015) "Perspective: Insight into reaction coordinates and dynamics from the potential energy landscape", *Journal of Chemical Physics, 142* (13).

Wales, D. (2003) *Energy landscapes: Applications to clusters, biomolecules and glasses*, Cambridge University Press, .

Wales, D. J. and Doye, J. P. (1997) "Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms", *The Journal of Physical Chemistry A, 101* (28), 5111–5116.

Wales, D. J. and Scheraga, H. A. (1999) "Global optimization of clusters, crystals, and biomolecules", *Science, 285* (5432), 1368–1372.

Zhang, H. and Rangaiah, G. P. (2011) “A Review on Global Optimization Methods for Phase Equilibrium Modeling and Calculations”, *The Open Thermodynamics Journal,* pages 71–92.

Žilinskas, J. (2008) “Branch and bound with simplicial partitions for global optimization”, *Mathematical Modelling and Analysis, 13* (1), 145–159 URL `http://dx.doi.org/10.3846/1392-6292.2008.13.145-159`.